SPAM SMS Detection

|  |  |
| --- | --- |
| Name | ID |
| Maruf Shahriar | 20-42016-1 |
| Asif Hossain Neloy | 20-42996-1 |
| Muhaiminul Ashrafee | 20-42217-1 |
|  |  |

**Abstract:**

We are currently living on the world of data and technology. Almost every aspect of our life is connected to a modern equipment or technology. The mobile devices are one of the best examples of this. It is a major source of information and communication. Though the necessary data we need is provided by the device, a lot of unnecessary and harmful data known as spam is slipped into the mix to harm or hurt the user in various ways. The proliferation of mobile devices and messaging applications has led to a rise in unsolicited SMS messages, commonly known as SMS spam. The spams SMS cause the user various kinds of problems including financial loss, identity theft, phycological impact, waste of device space and resource. In the recent years it has become a major issue. The increase in SMS spam has prompted the development of machine learning techniques to detect and filter out such messages. In this research paper, we explore the application of machine learning algorithms to identify and classify SMS spam messages. We propose a models that uses text pre-processing techniques to extract features from SMS messages, and then applies a range of supervised machine learning algorithms to classify messages as spam or not spam. We evaluate the performance of our models using a publicly available dataset of SMS messages, and compare it against several baseline modelss. Our results show that our proposed models outperforms the baseline modelss, achieving an accuracy of over 95% in detecting spam messages. We also conduct an analysis of the features that contribute most to the classification of spam messages, and find that certain keywords and phrases are highly indicative of spam messages. Our findings can be used to develop more effective SMS spam filters and improve the overall user experience of mobile messaging applications.

**1. Objective:**

The objective of building an SMS-spam detection machine learning models is to accurately identify whether an incoming SMS message is legitimate or spam. The models is trained on a dataset of SMS messages that have been labeled as either legitimate or spam. The objective of the models is to learn the patterns and features of each message type and use this knowledge to classify new, unseen messages as either legitimate or spam. The ultimate goal of SMS-spam detection is to help prevent unwanted messages from reaching users' phones. This can help to reduce the number of unwanted marketing messages, phishing attempts, and other types of spam that users receive on a daily basis [1]. By accurately detecting spam messages, the models can also help to improve the user experience and protect users from potential security risks.

**2. Methodology:**

This study aims to introduce a machine learning-based models that will filter spam SMS. With the help of machine learning algorithms, spam detection modelss will be created from the training data then spam detection will be made on the test data. In our study Logistic Regression, Support Vector Machine, Naïve Bayes, Decision Tree, KN Classifier, Randomforest Classifier, GradientBoosting Classifier, ExtraTrees Classifier, Bagging Classifier, AdaBoost Classifier, XGB Classifier algorithms are used. Feature selection and data transformation methods will be applied to the data to increase spam detection success. For feature extraction we have chosen TfidfVectorizer method.

**2.1 Data Collection:**

Our goal is to filter out spam SMS from important SMS. To do so, we need data that we can work on. The first step in data collection for SMS-spam detection is to obtain a representative sample of SMS messages from various sources, such as SMS service providers, online forums, and social media platforms. The messages should be a mix of spam and legitimate messages to ensure that the models can accurately distinguish between the two. We have chosen the dataset named spam which is publicly available at Kaggle [2]. The dataset contains 5572 rows and 5 columns. An important consideration in data collection for SMS-spam detection is privacy. To ensure that any personal information contained within the messages is handled in a secure and ethical manner. Through anonymizing the data and changing the name of the users, we have stopped any possible way of privacy breach. The data contains 4516 SMS that are valid and 653 SMS that are spam. The data contains text, numbers, special characters, empty spaces. We have dedicated 80% of the data as training data and 20% of the data as testing data to measure accuracy and precision.

**2.2 Data Validation:**

Data validation is a process of ensuing that the data is correct and up to the standard to be used in a research scenario. Valid and correct data is the backbone of any training models. Almost every data set contains unnecessary and undesired data. The use of these data without processing and sorting them out will result in poor accuracy. A training models will be able to achieve higher level of accuracy and precision if the data validation is done properly. In our dataset, we have performed these steps to validate the data.

1. Data cleaning: In this step, we have observed that 3 out of the 5 columns are mainly empty. So, we have dropped the 3 unnecessary columns.

2. EDA: We have sorted the spam and ham SMS in table and assign ham as 0 and spam as 1 in the table.

3. Text Preprocessing: Checking for null values, duplicate values and removing them.

4. Making Correlation: We have calculated number of characters, words and sentences and established a relationship between these features and spam or ham SMS.

5. Splitting the dataset: The dataset is split into two as train and test datasets having 80% and 20% of the data respectively.

**2.3 Data Preprocessing:**

Data preprocessing takes a deeper look into the data and removes inconsistent and non-useable values form the data. This step hammers out further uncertainty and improves the quality and reliability of the data by a significant margin. The preprocessing contained the following steps.

1.Lower case: Convert all texts into lower case lower the variation between data.

2.Tokenization: Breaking data into smaller bits so that it is easier to process.

3.Removing special characters: Remove any special characters that might be in the dataset.

4.Removing stop words and punctuation: Remove punctuations from the dataset as they do not convey meaning or context in this scenario.

5.Stemming: This indicates making the variations of similar words the same. For example: walks, walked, walking will all be converted into walk.

**2.4 Feature Extraction:**

The purpose of feature extraction is to select and extract features from SMS. Here we make sure the data are sorted and labeled according to the values we need. The reason for labeling is to make it easier to apply different boundaries [3]. Here the dataset initially had 5 columns and 5572 rows. After data validation and preprocessing we were left with 2 columns named ham and spam. We then set the value 0 to ham and the value 1 to spam. After separating the data in X\_Test, Y\_Test, X\_Train, Y\_Train we use the TfidfVectorizer function to transform text data into numerical value or feature vector These feature vectors or numerical values will be thrown as inputs for our models. The function will generate score and those scores will be mapped with spam label.

**2.5 Classification Algorithm:**

Logistic Regression: The logistic regression statistic modelsing technique is used when we have a binary outcome variable. Though we may have continuous or categorical independent variables, we can use the logistic regression modelsing technique to predict the outcome when the outcome variable is binary. Logistic regression predicts categorical outcome variables. Linear regression models regression line is highly susceptible to outliers.

Support Vector Machine: It is a supervised learning machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems, such as text classification. A simple linear SVM classifier works by making a straight line between two classes. This means there can be an infinite number of lines to choose from. What makes the linear SVM algorithm better than some of the other algorithms is that it chooses the best line to classify your data points. It chooses the line that separates the data and is the furthest away from the closet data points as possible.

Naïve Bayes: Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning modelss that can make quick predictions [4]. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Decision Tree: Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

KN Classifier: K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

Randomforest Classifier: Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

GradientBoosting Classifier: Gradient Boosting is a functional gradient algorithm that repeatedly selects a function that leads in the direction of a weak hypothesis or negative gradient so that it can minimize a loss function. Gradient boosting classifier combines several weak learning modelss to produce a powerful predicting models. Loss function, Weak Learner and Additive Models are the weak modelss that are included in Gradient Boosting Classifier [5].

ExtraTrees Classifier: The extra trees algorithm, like the random forest algorithm, creates many decision trees, but the sampling for each tree is random, without replacement. This creates a dataset for each tree with unique samples. A specific number of features, from the total set of features, are also selected randomly for each tree.

Bagging Classifier: A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregates their individual predictions to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator, by introducing randomization into its construction procedure and then making an ensemble out of it.

AdaBoost Classifier: AdaBoost, also called Adaptive Boosting, is a technique in Machine Learning used as an Ensemble Method. The most common estimator used with AdaBoost is decision trees with one level which means Decision trees with only 1 split. These trees are also called Decision Stumps.

XGB Classifier: XGBoost is a robust machine-learning algorithm that can help understand data and make better decisions. XGBoost is an implementation of gradient-boosting decision trees. It has been used by data scientists and researchers worldwide to optimize their machine-learning modelss.

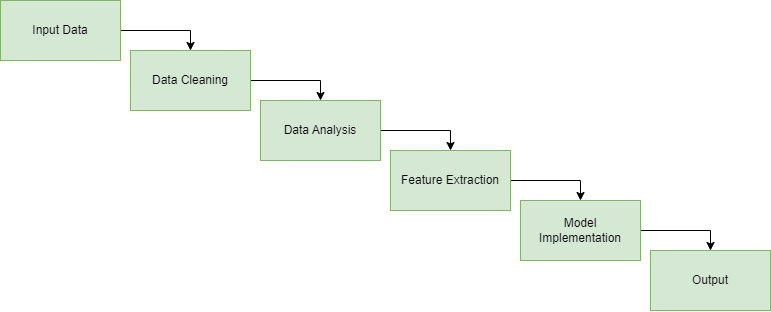
These are the classification algorithms that are used. All of these classification algorithms are trained and tested and we have measured the outcome of each algorithm based on accuracy and precision.

**2.6 Data Analysis:**

In our models, we have followed quantitative data analysis technique. We have previously categorized spam and non-spam mail values as 0 and 1. We have also converted the columns into numeric scores and mapped them into numeric labels. So, we can see that the majority of the data and process requires working with data. Thus, quantitative or numeric data analysis is the correct data analysis technique for our models.

**2.7 Block Diagram:**

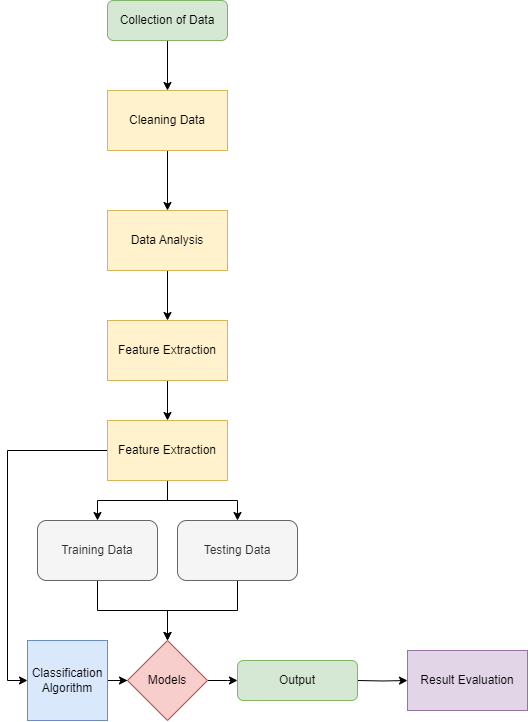
The block diagram for our models is given below:



*Figure 1: Block Diagram*

**2.8 Workflow Diagram:**

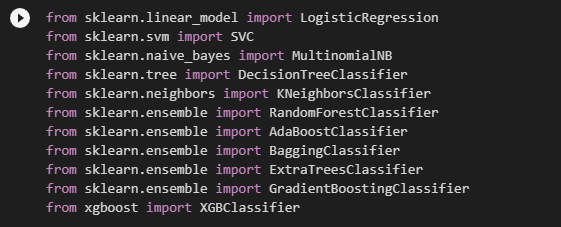
The workflow diagram for our models is given below:



*Figure 2: Workflow Diagram*

**3. Experimental Setup and Implementations:**

Our models requires some libraries. Some of them are given below-



*Figure 3: Importing Libraries*

Then we can import the dataset from content section in the environment



*Figure 4: Importing Dataset*

While implementing the data, we can start with the feature extraction



*Figure 5: Feature Extraction*

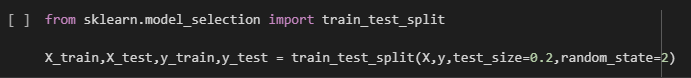
Now we can set the columns into X and Y





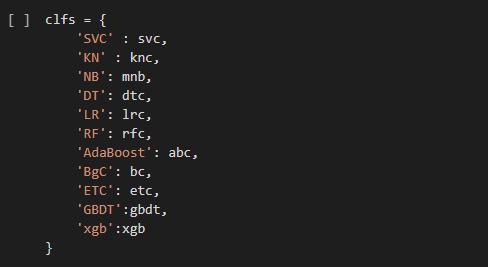
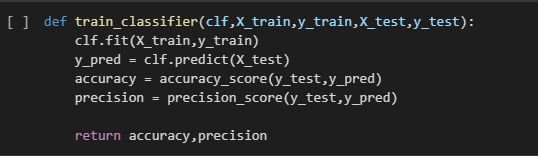
*Figure 6: Making X and Y array*

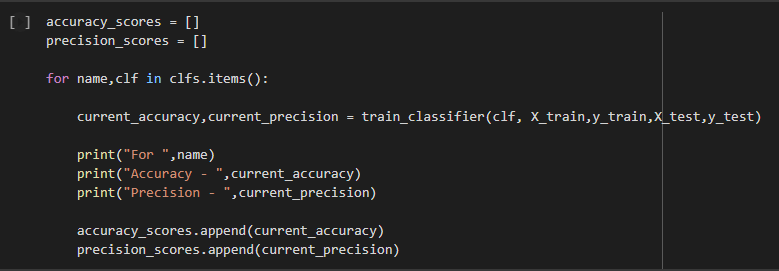
After that we can split the data into 2 sections for train and test



*Figure 7: Splitting Dataset*

Then we can run the algorithms and find the accuracy and precision values





*Figure 8: Implementing and Running Algorithms*

**3. Result and Discussion:**

The result for the modelss is measured by precision and accuracy. We have multiple algorithms and the results of them are

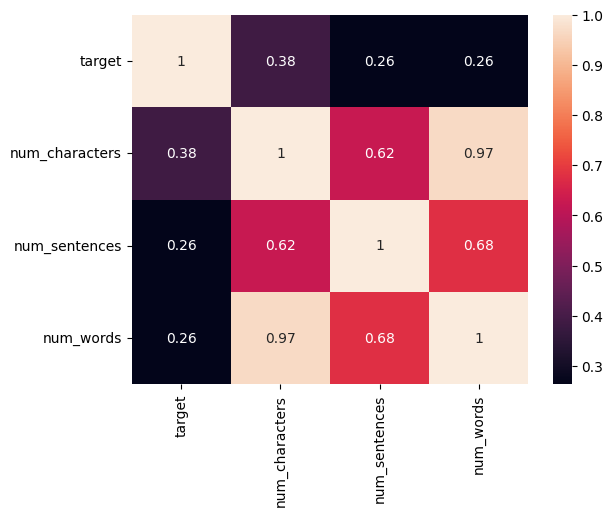
|  |  |  |
| --- | --- | --- |
| Algorithm | Accuracy | Precision |
| KN Classifier | 0.900387 | 1.000000 |
| Naïve Bayes | 0.959381 | 1.000000 |
| Randomforest Classifier | 0.971954 | 1.000000 |
| ExtraTrees Classifier | 0.972921 | 0.982456 |
| Support Vector Machine | 0.972921 | 0.974138 |
| AdaBoost Classifier | 0.961315 | 0.945455 |
| Logistic Regression | 0.951644 | 0.940000 |
| XGB Classifier | 0.970019 | 0.934959 |
| GradientBoosting Classifier | 0.952611 | 0.923810 |
| Bagging Classifier | 0.958414 | 0.862595 |
| Decision Tree | 0.935203 | 0.838095 |

**3.1 Result Comparison:**

From the result table we can see the values of accuracy and precision. Our main focus is to stop false positives while making sure that the accuracy of the models is high. So, we are going to look for high precision and high accuracy. So, from the table, the best performing algorithm in this case is Randomforest classifier with the highest sum of precision and accuracy. The worst algorithm to be used in this case is Decision Tree with lowest precision and low accuracy.

|  |  |  |  |
| --- | --- | --- | --- |
| Decision | Algorithm | Accuracy | Precision |
| Best Choice | Randomforest Classifier | 0.971954 | 1.000000 |
| Worst Choice | Decision Tree | 0.935203 | 0.838095 |

**3.2 Confusion Matrix Analysis:**

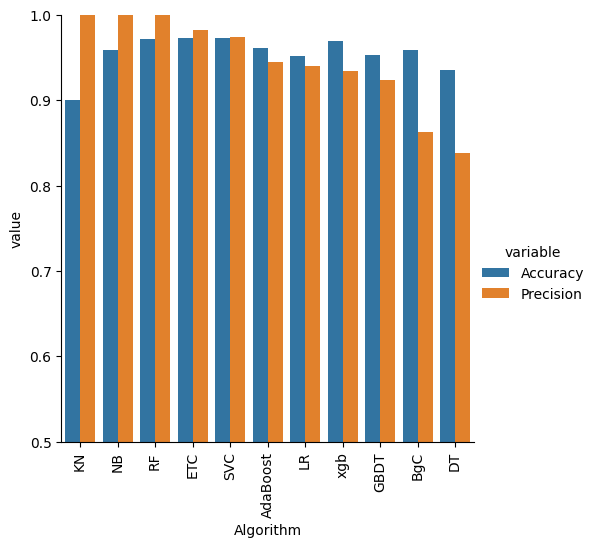


*Figure 9: Heat Map/Confusion Matrix*

Here is a confusion matrix contains num\_characters, num\_sentences, num\_words and target plotted as a heatmap. This confusion matrix can help us to understand the correlation between each element situated in this heat map.

**3.2 Graphical Representation of Results:**

The visual representation of the data is shown below:



*Figure 10: Heat Map/Confusion Matrix*

The results here is represented as a bar graph and along the x and y axis there is Algorithms used and the values of accuracy and precision given in different color.

**4. Conclusion and Future Recommendations**

As a conclusion, we can see that among the algorithms that were used here, Randomforest Classifier was the preferred choice and had the best outcome. But if we change the size of dataset or number of features used, other algorithms will provide a better result that might even surpass our current best choice.

As future recommendation, we can use the extracted features num\_characters, num\_sentences, num\_words as part of the dataset and see if that will improve the outcome. Adding extra data that have relationships with dataset could be beneficial.

**5. References**

[1] V. A. Saeed, “A method for SMS SPAM message detection using machine learning,” *Artificial Intelligence & Robotics Development Journal*, pp. 214–228, 2023.

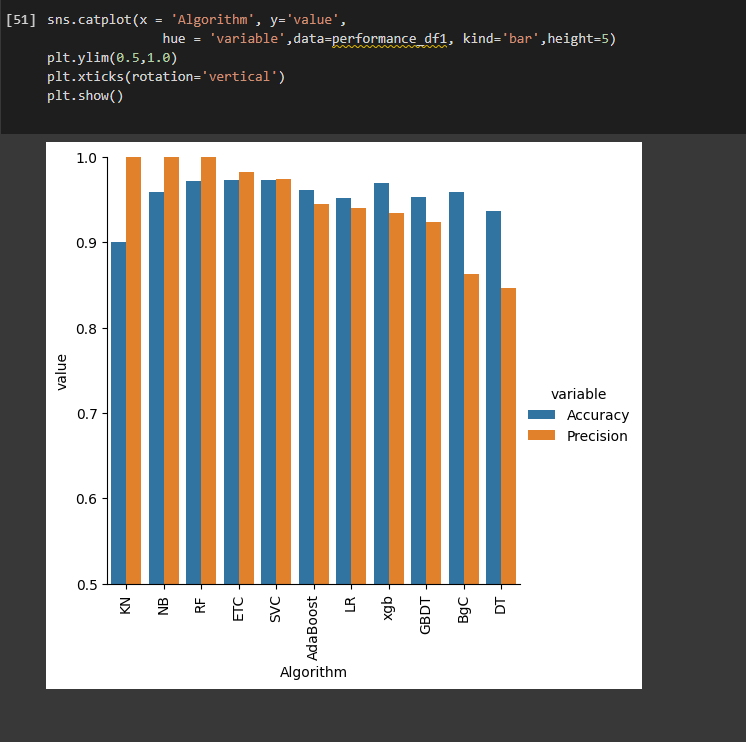
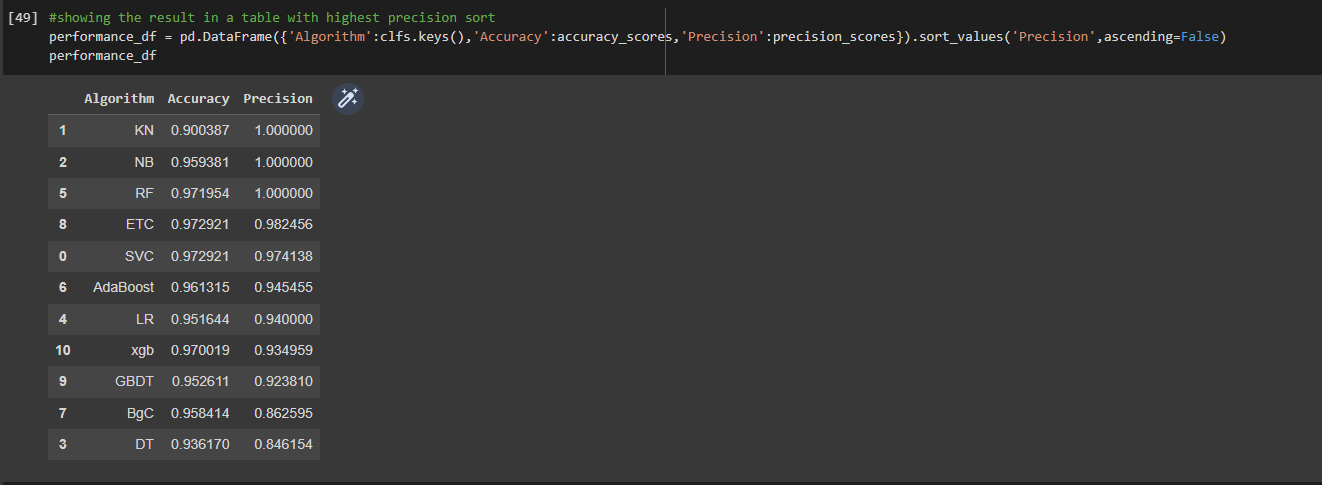
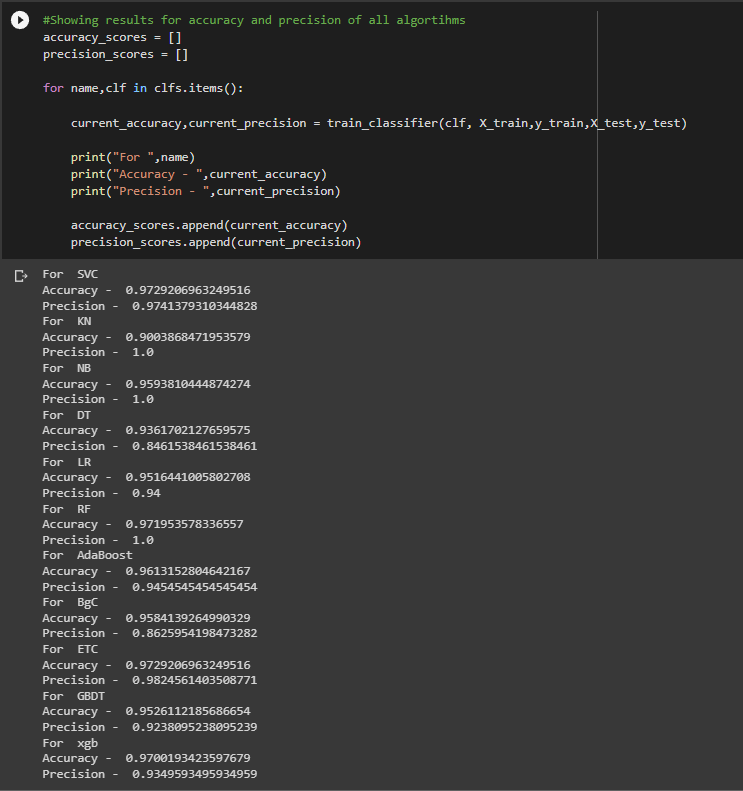
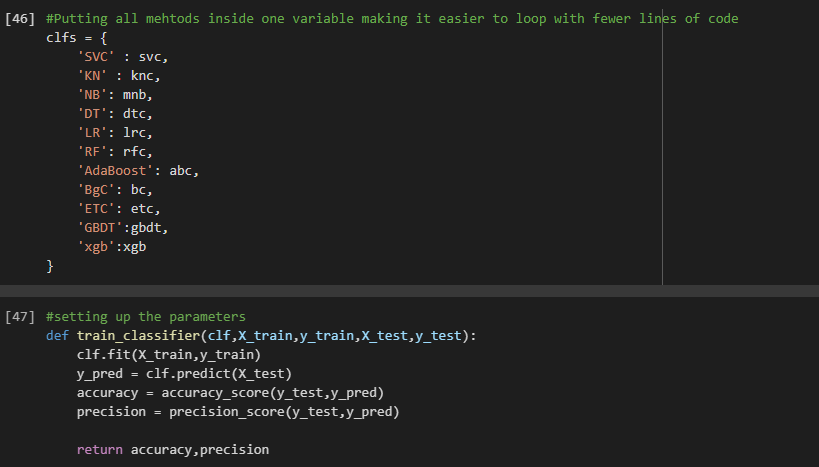
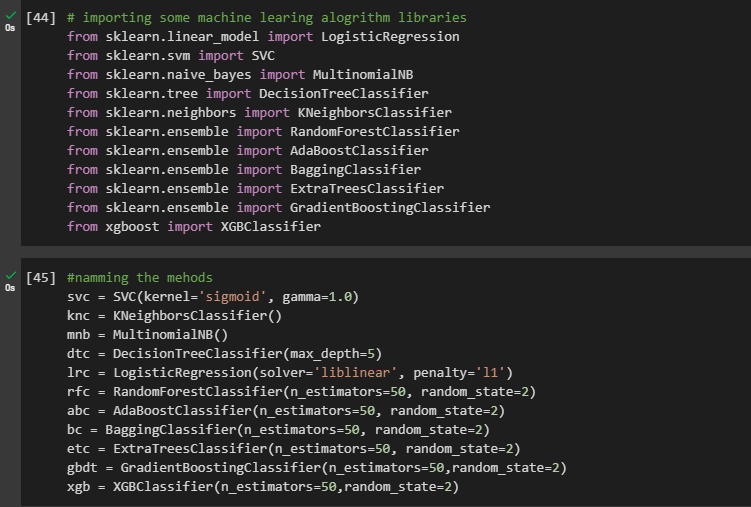
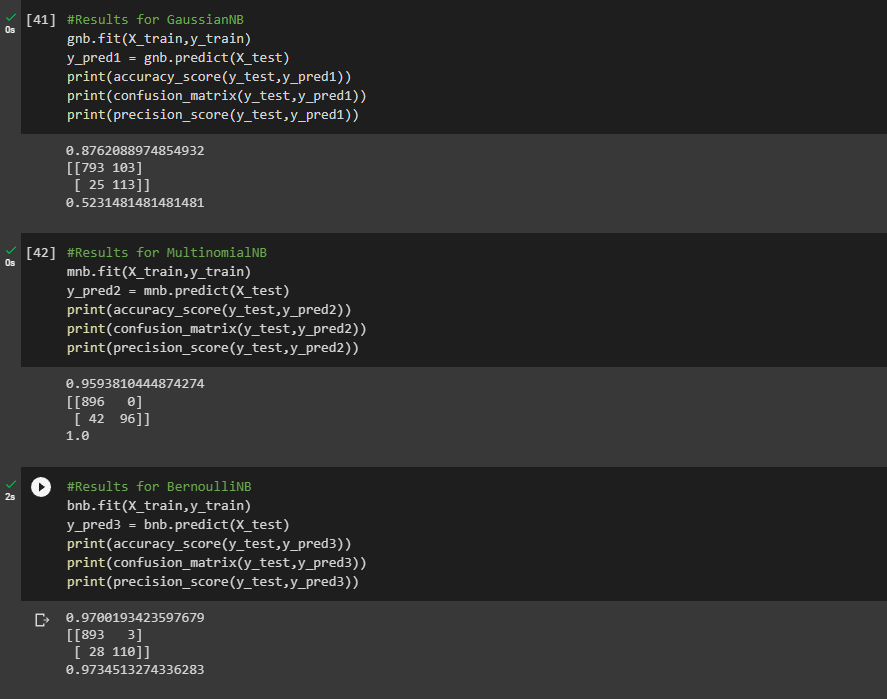
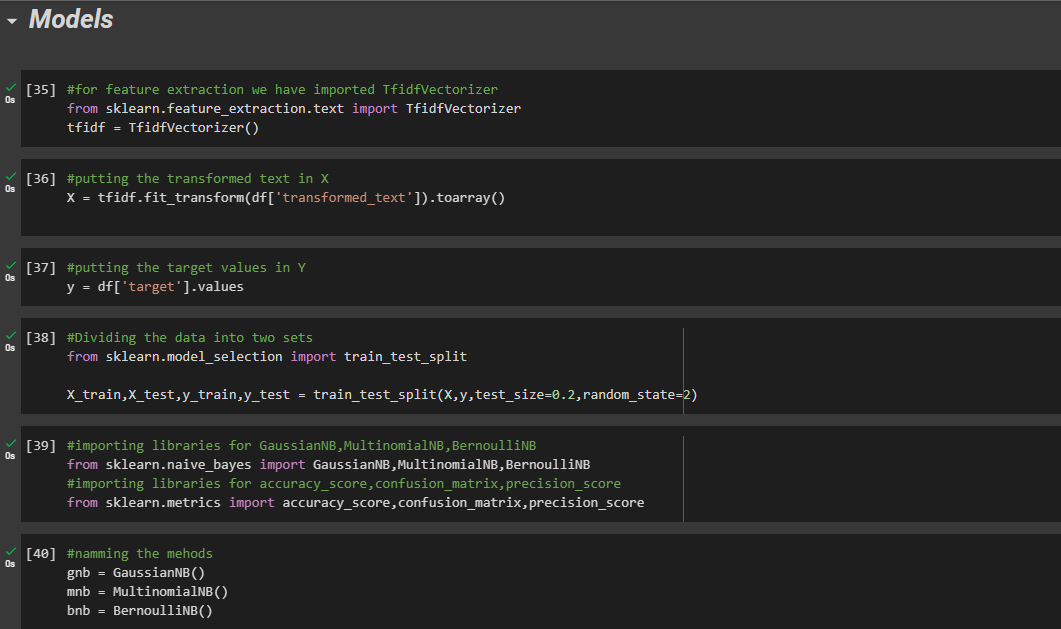
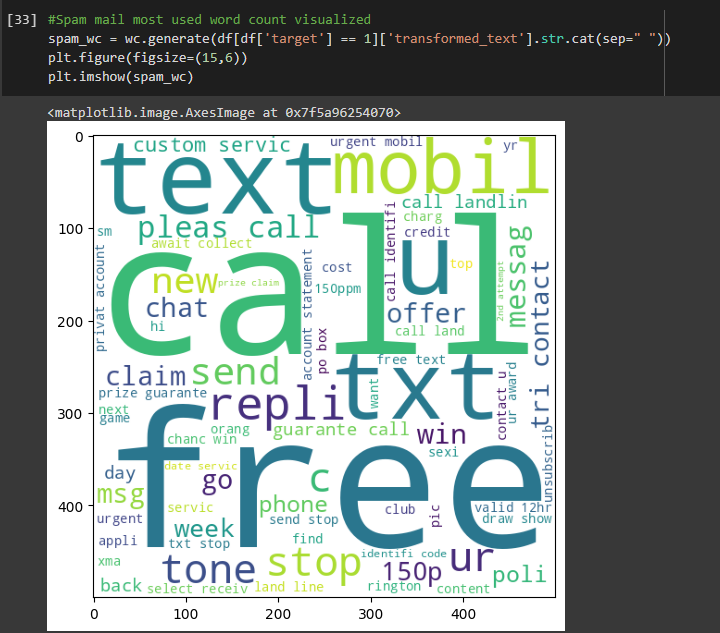
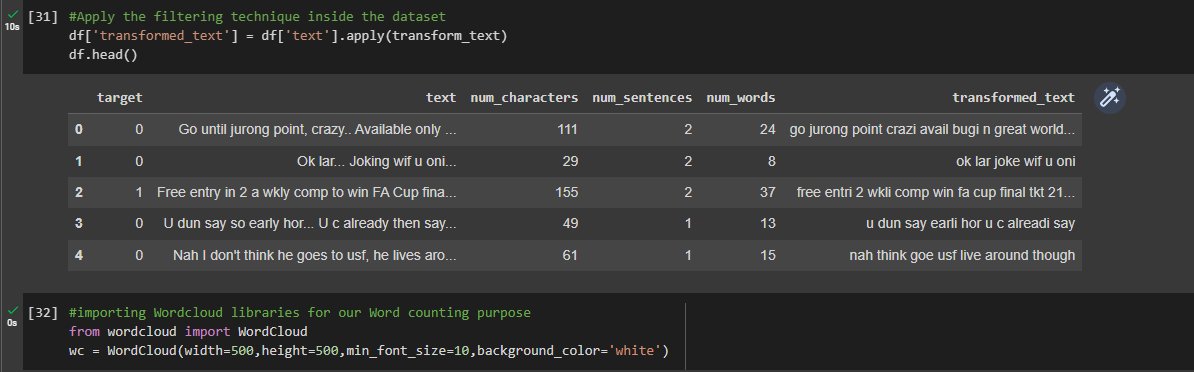
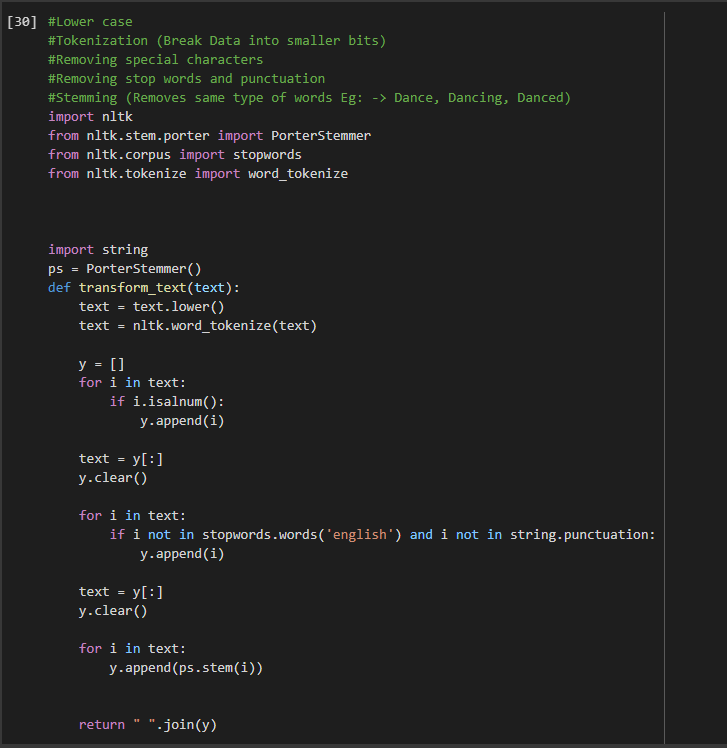
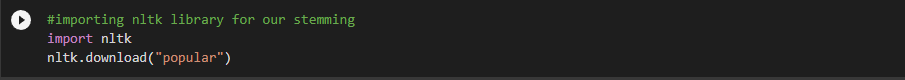
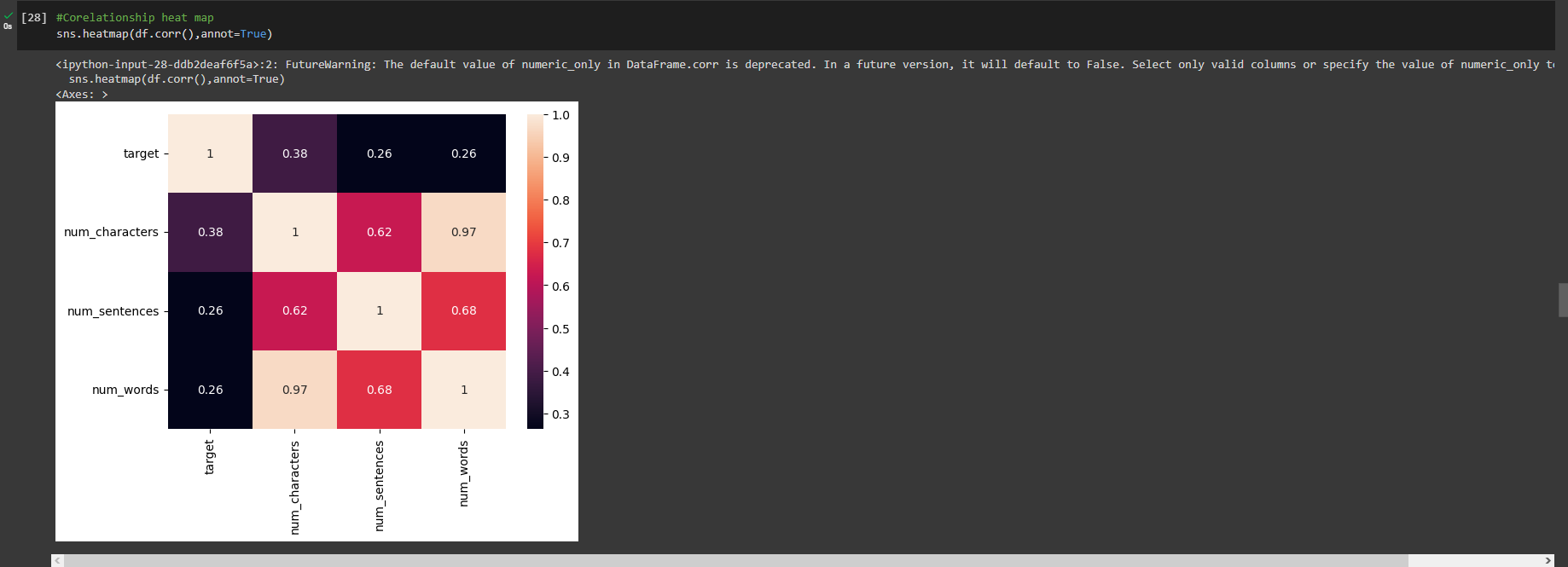
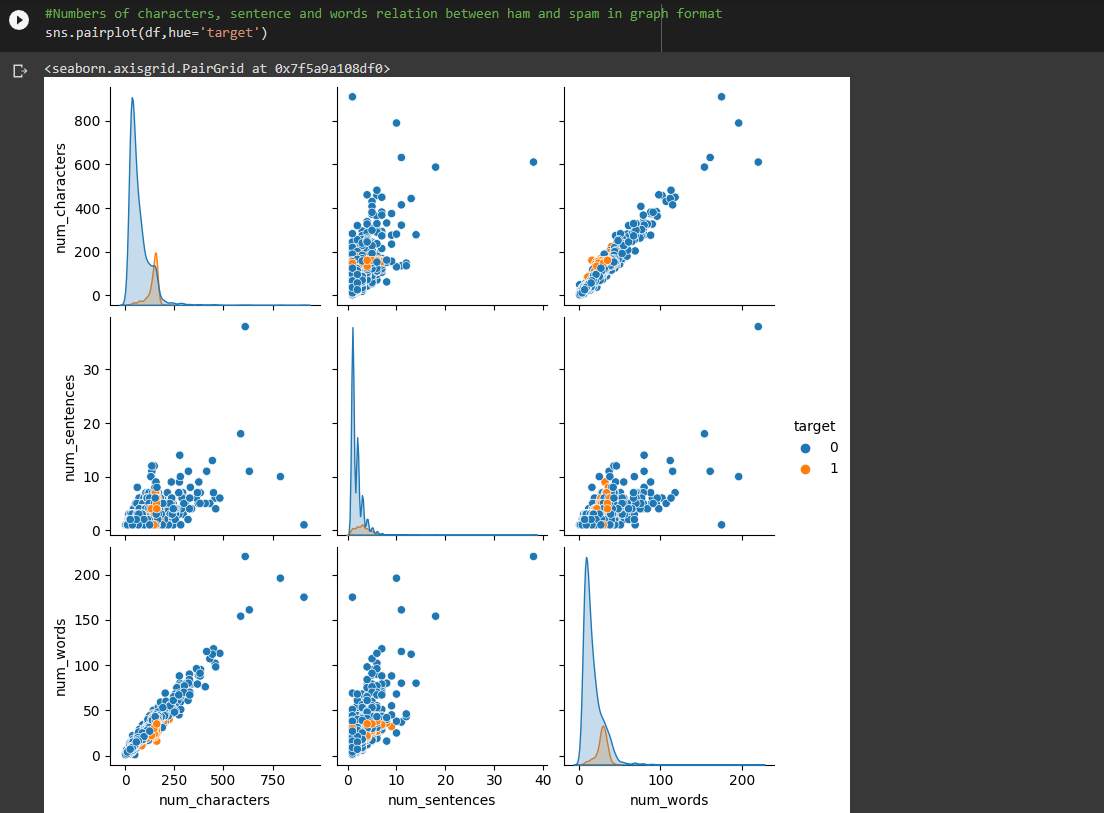
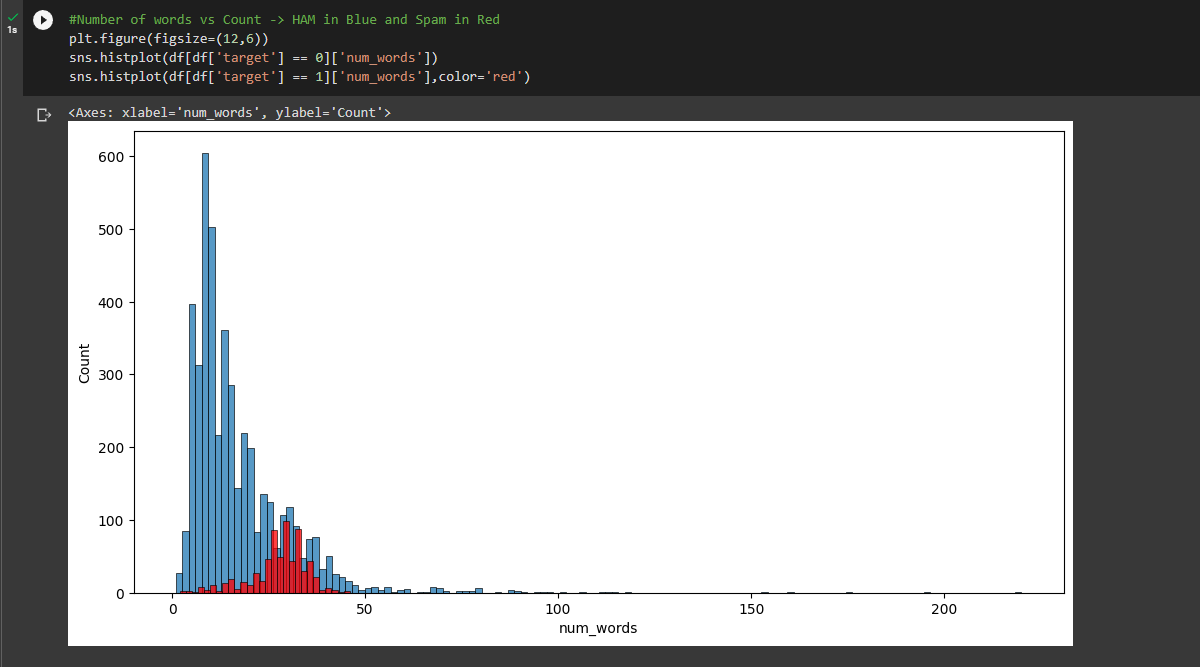
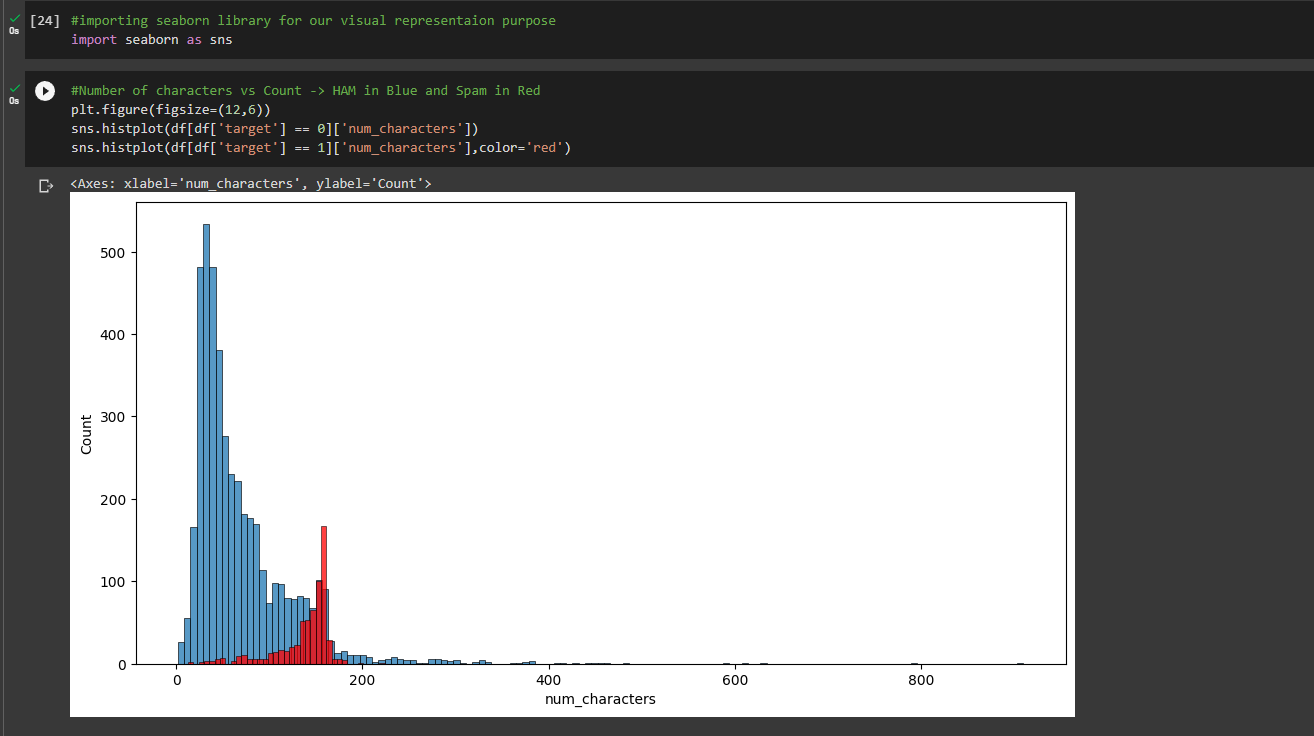
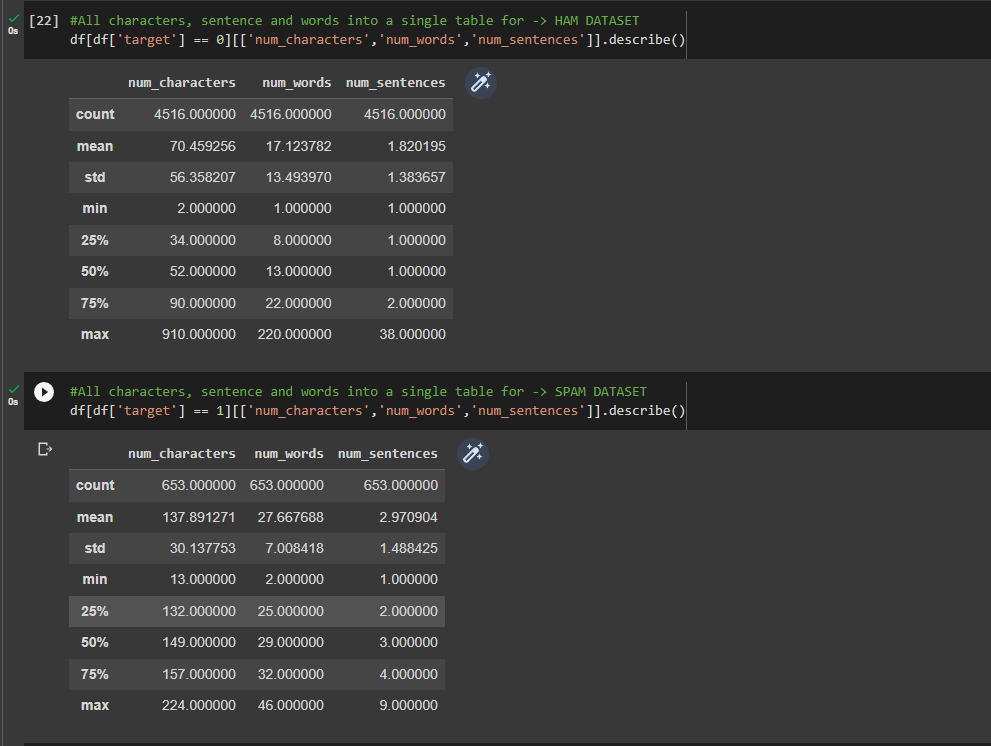
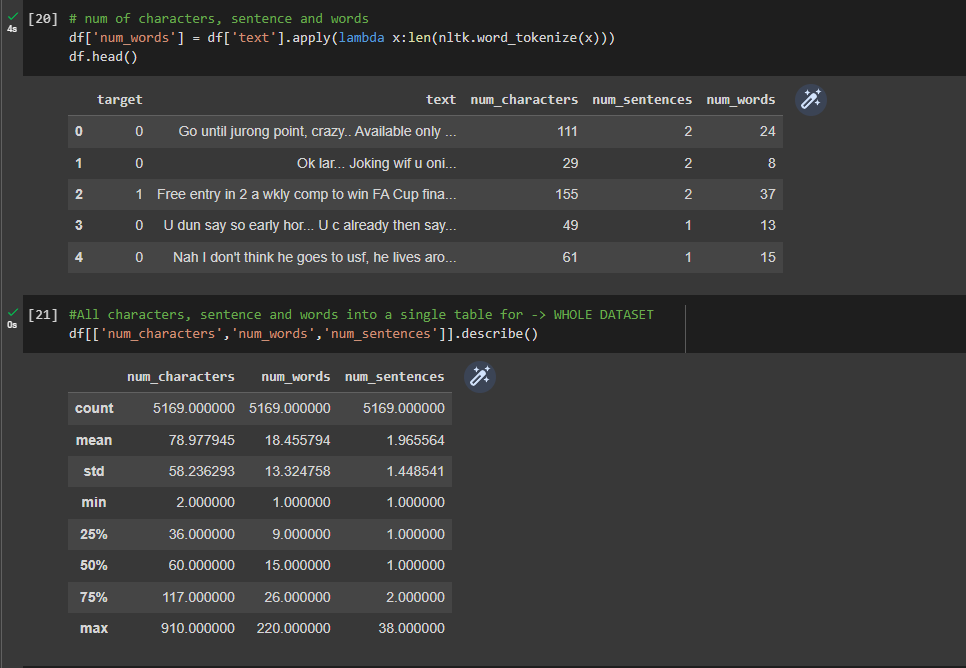
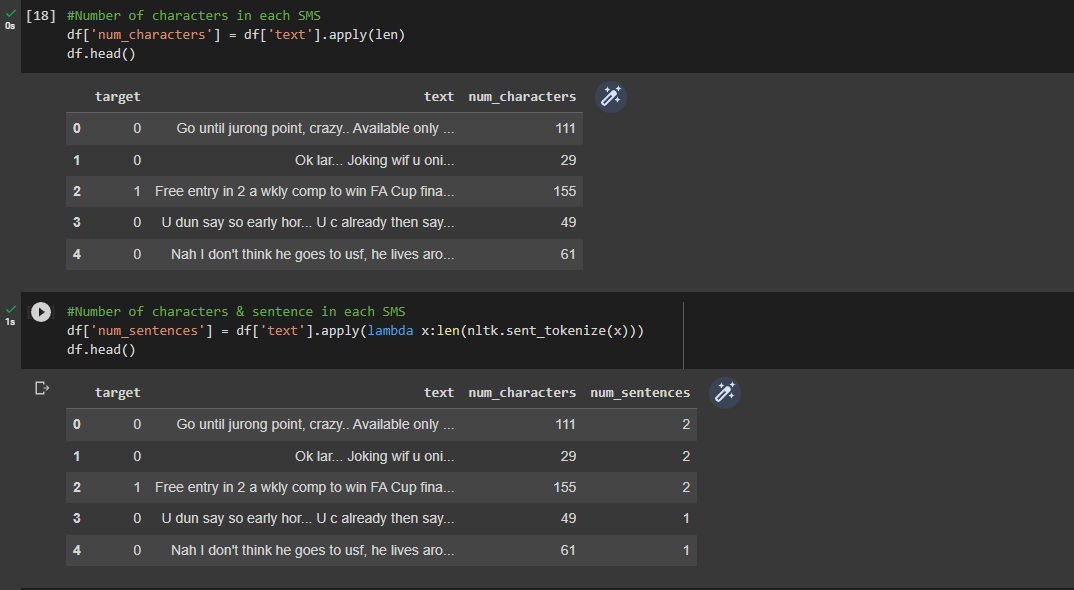
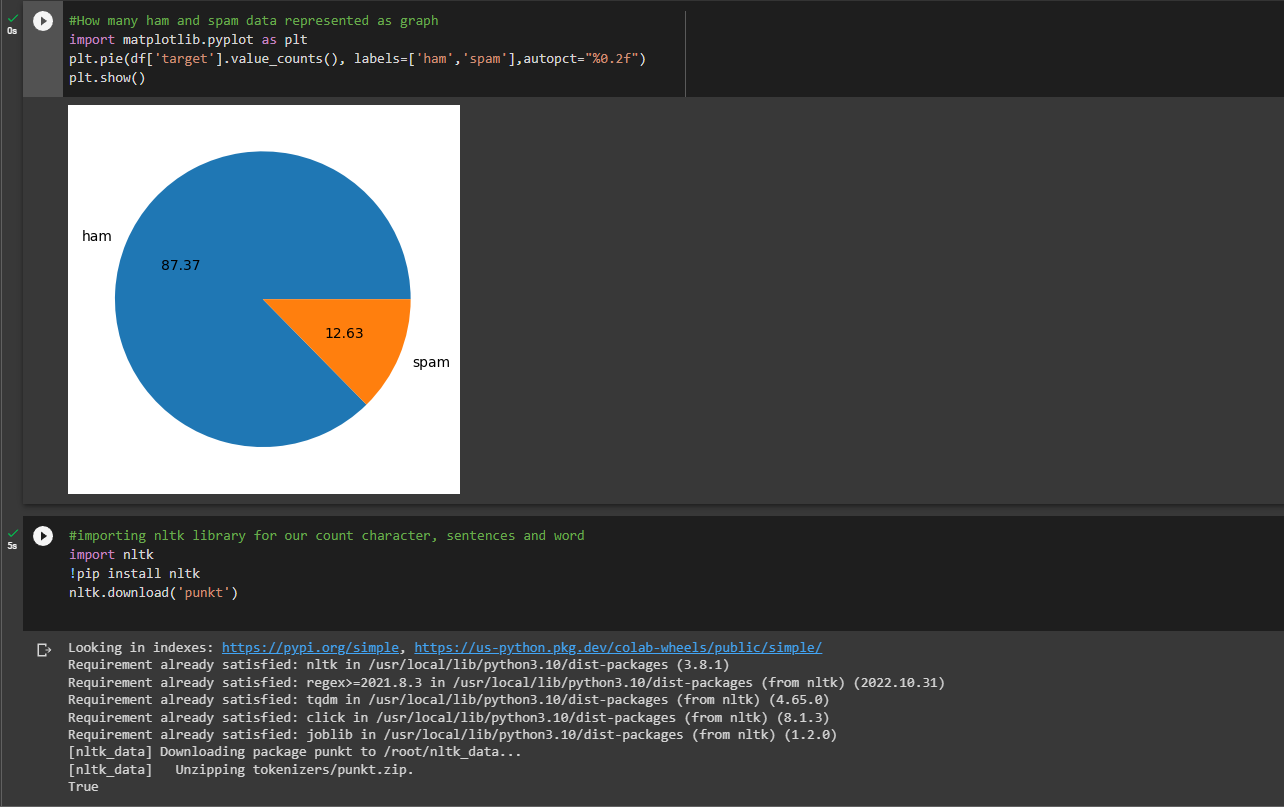
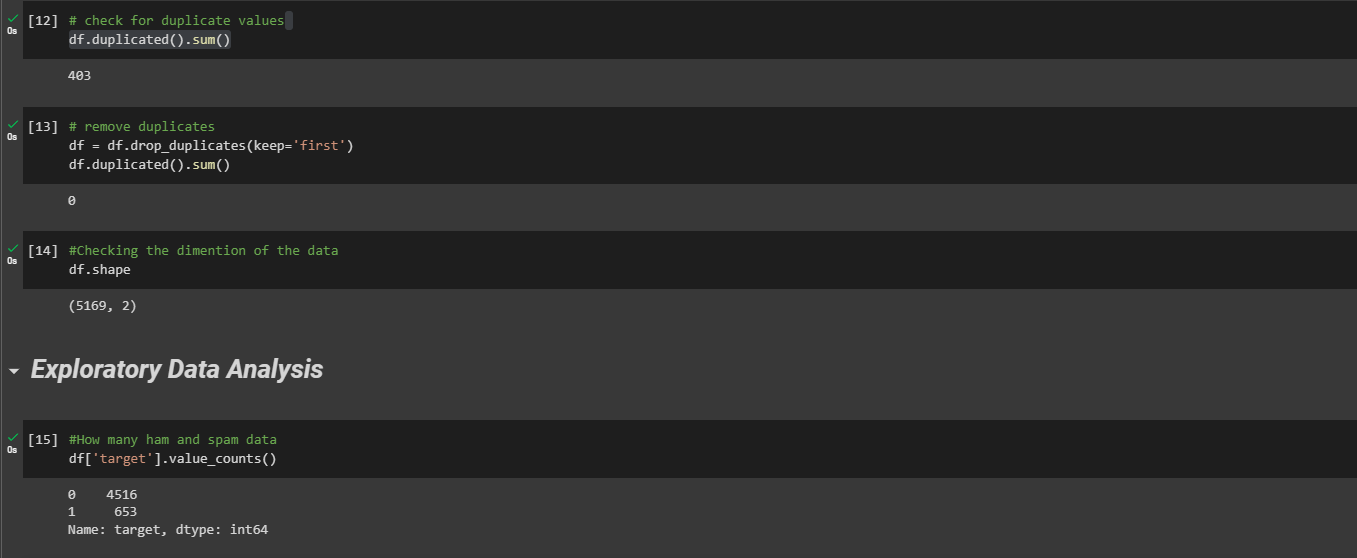
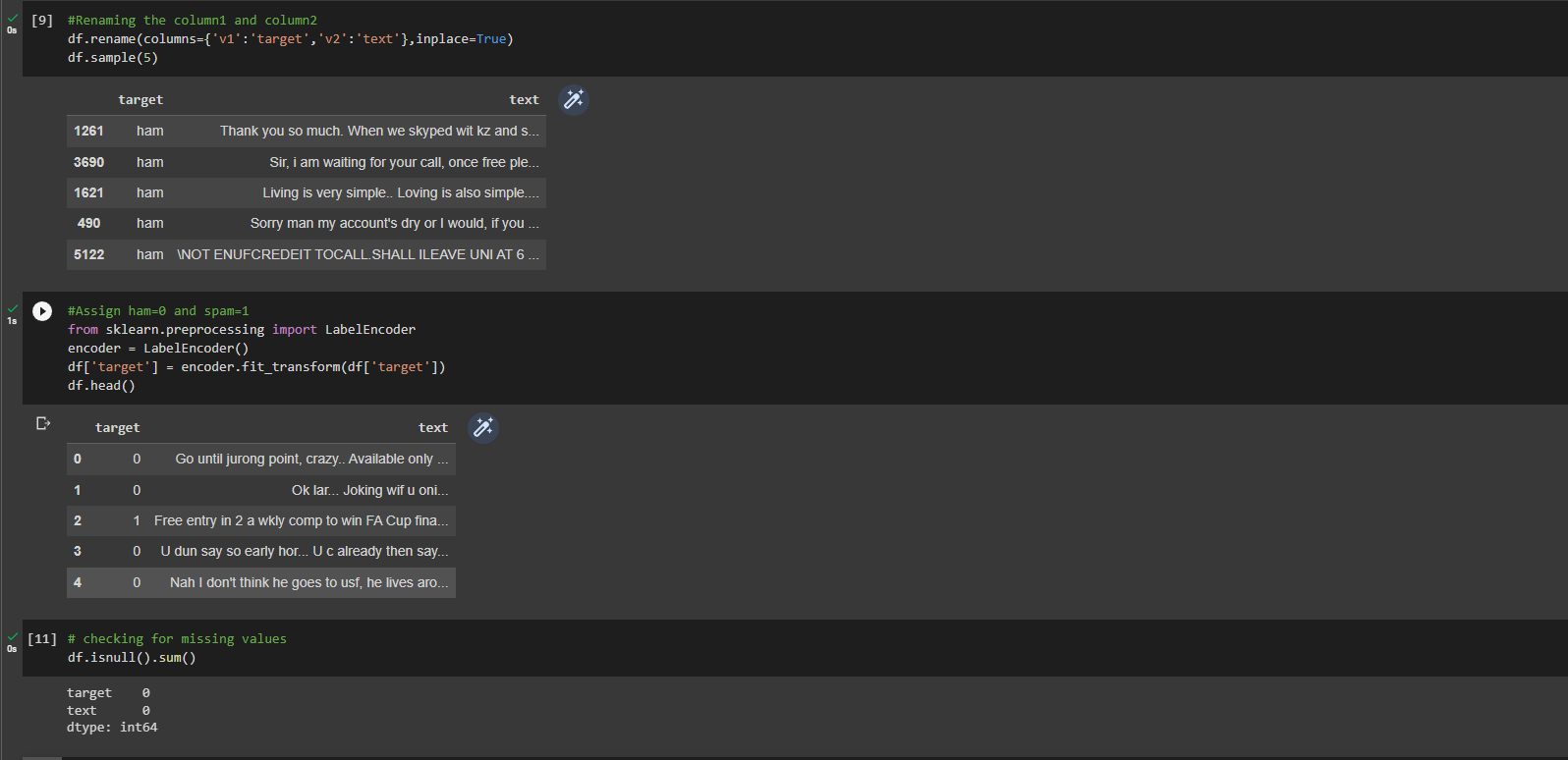
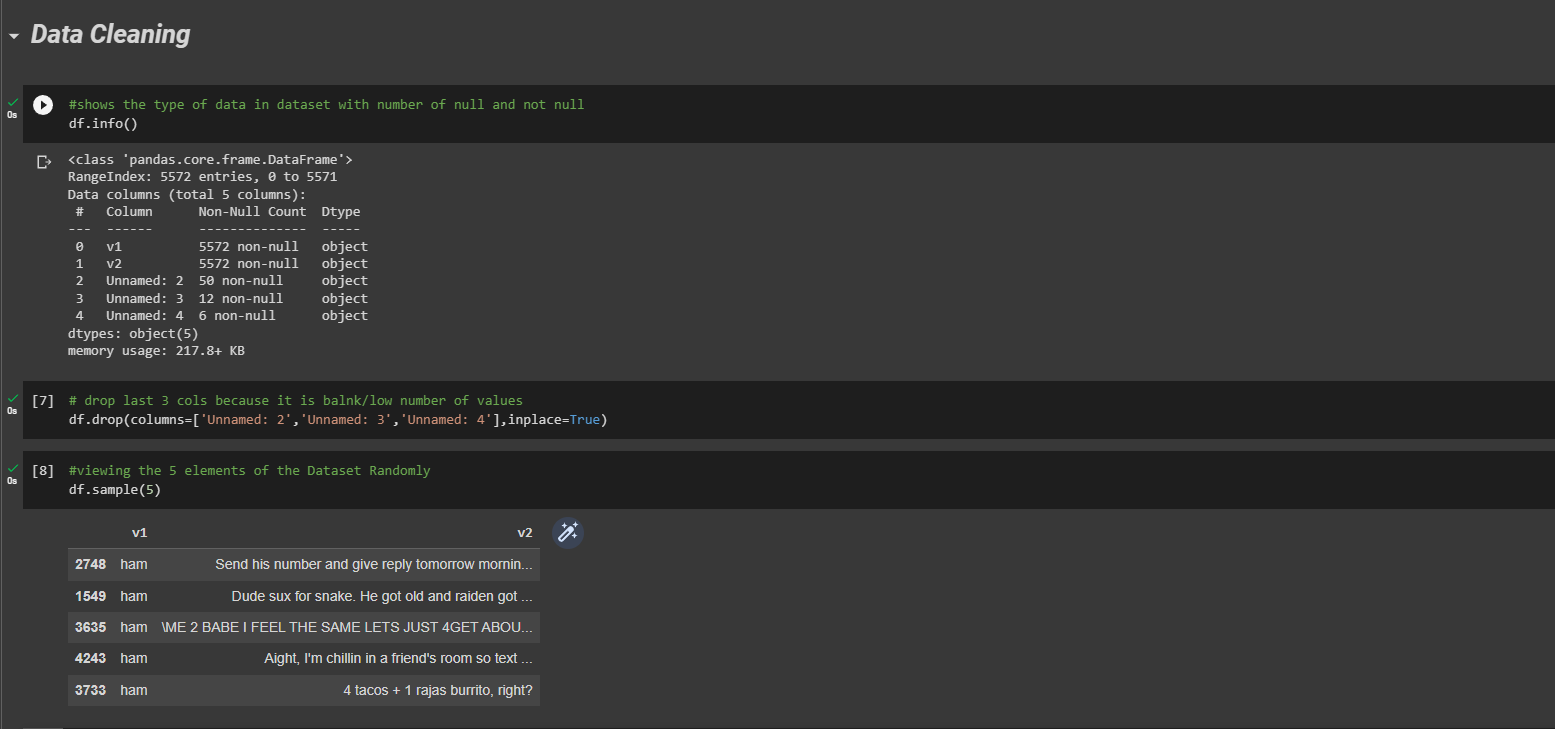
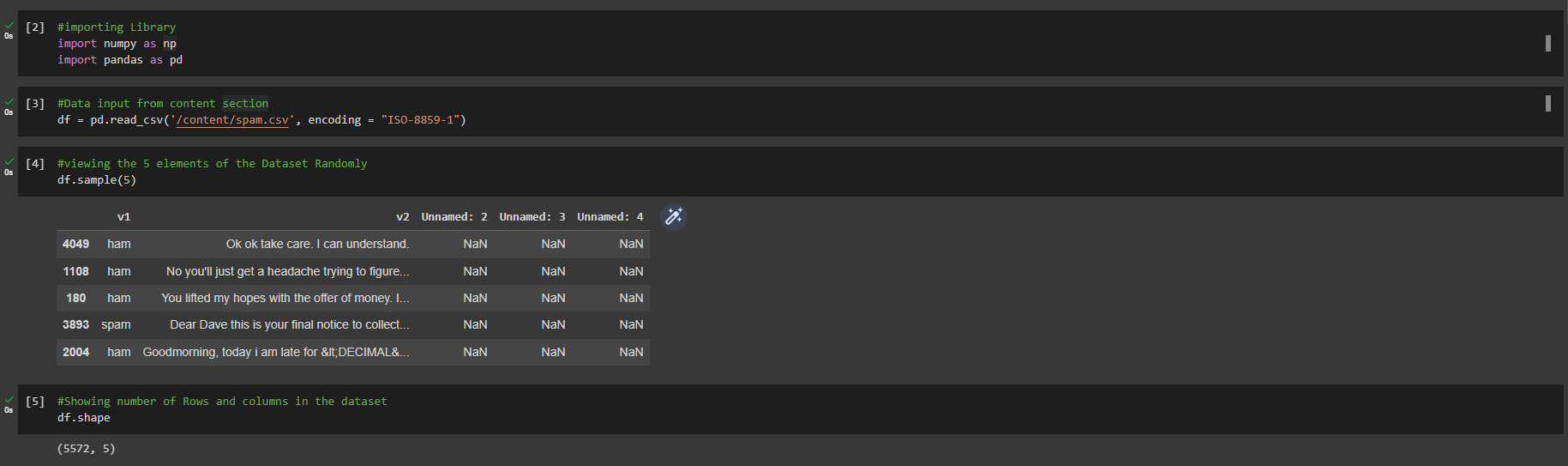
[2]U. C. I. M. Learning, “SMS Spam Collection Dataset,” *Kaggle*, 02-Dec-2016. [Online]. Available: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset. [Accessed: 03-May-2023].

[3] S. Gadde, A. Lakshmanarao, and S. Satyanarayana, “SMS SPAM detection using machine learning and Deep Learning Techniques,” *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2021.

[4] M. Gupta, A. Bakliwal, S. Agarwal, and P. Mehndiratta, “A comparative study of spam SMS detection using machine learning classifiers,” *2018 Eleventh International Conference on Contemporary Computing (IC3)*, 2018.

[5] A. K. Jain, S. K. Yadav, and N. Choudhary, “A novel approach to detect spam and smishing SMS using machine learning techniques,” *Research Anthology on Securing Mobile Technologies and Applications*, pp. 267–285, 2021.

**Appendixes:**

****