International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Spam SMS Classification

[1]Ms. Akshara Prachi, [2]Mrs. Ankita Gandhi, [3]Mr. Garv Kavdia, [4]Mr. Deepak Parmar

[1,2,3,4] Department of Computer Science & Engineering, Parul University Vadodara, Gujarat, India
[1]akshara.jha29428@paruluniversity.ac.in, [2]ankita.gandhi@paruluniversity.ac.in, [3]210303105500@paruluniversity.ac.in,
[4]210303105200@paruluniversity.ac.in

A B S T R A C T

Spam SMS messages pose a significant security and privacy risk to the users, and therefore effective detection methods are needed. Spam messages normally bear malicious links, phishing, or unwanted advertisements, and hence filter mechanisms need to be established. The present work focuses on creating an NLP-based machine learning spam classifier. The studies examine various classification models, e.g., Naïve Bayes,

Support Vector Machines (SVM), and Random Forest, how their performance with respect to accuracy, precision, recall, and F1-score. Our model is implemented using Python libraries such as Scikit-learn, Pandas, NumPy, and NLTK, and automation using Selenium for dataset collection. The results indicate that our model achieves an accuracy of 97%, which significantly improves spam detection efficiency. The results indicate that appropriate preprocessing, model choice, and feature extraction are crucial in spam classification. Future work will emphasize incorporating deep learning methods and cloud deployment to further enhance system performance.

Keywords: Spam SMS classification, machine learning, Natural Language Processing (NLP), Naïve Bayes, Support Vector Machine (SVM), Random Forest, feature extraction, Term Frequency Inverse Document Frequency (TF-IDF), web scraping, automation, Selenium, data preprocessing, text classification, supervised learning, classification models, model evaluation, precision, deep learning, cloud-based deployment.

Cite This Article As: Akshara Prachi, Ankita Gandhi, Garv Kavdia, Deepak Parmar (2025). Spam SMS Classification International Journal of Research Publication and Reviews.

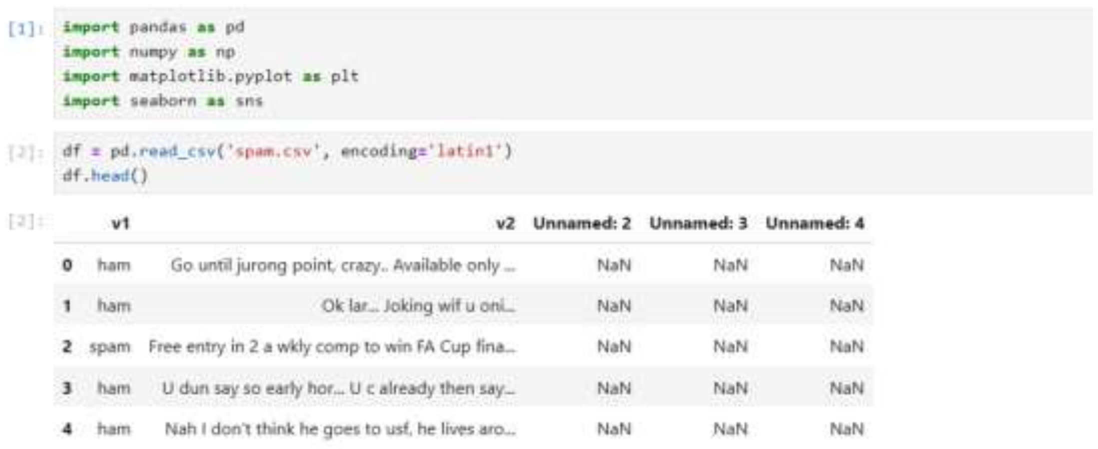
1.0 Introduction

With greater use of mobile communication, spam messages have become an issue of interest among telecom operators and users as well. Not only do they cause inconvenience, but they even pose threats towards security in the form of identity theft and money fraud. Traditional rule-based filter schemes cannot handle evolving spamming methods and are therefore machine learning is the most suitable remedy. With the use of natural language processing and supervised learning, the research work here aims at developing an automated spam filter system to identify SMS messages. The objectives within this study include the comparative evaluation of multiple classification frameworks, refinement of feature extraction methods, and providing a proper preprocessing pipeline to enhance the capabilities of detecting spam. Automation of data collection and processing is also discussed in developing model adaptability.

2.0 Methodology

Our approach consists of four main phases:

• Data Collection: We used Selenium and web-scraping techniques to collect SMS datasets from various sources, including public repositories, telecom datasets, and user-generated messages. The dataset consists of real-world SMS messages labelled as spam or ham (legitimate messages). Model generalization was further improved using additional data augmentation techniques, including the generation of synthetic data and balancing the data. Additionally, Selenium was used to automate the scraping of data from various websites, helping us collect a diverse and up-to-date dataset. The dataset preprocessing stage involved filtering out duplicate entries, removing irrelevant content, and structuring the data for efficient processing.

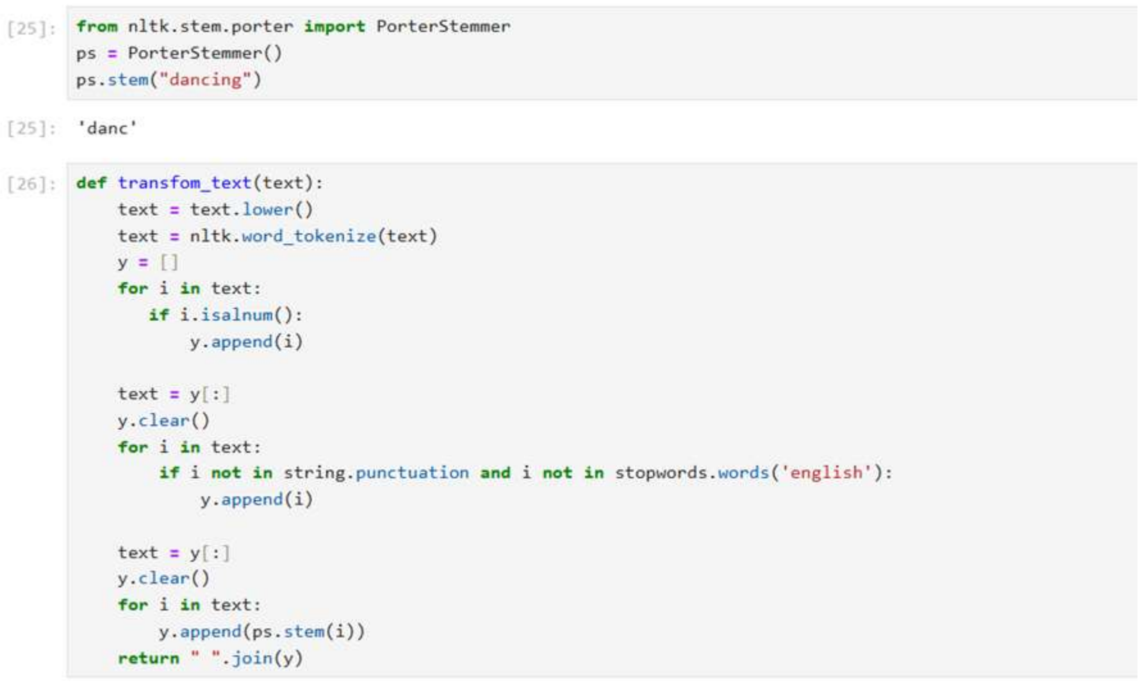


```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns

[2]: df = pd.read_csv('spam.csv', encoding='latin1')
     df.head()
```

[2]:

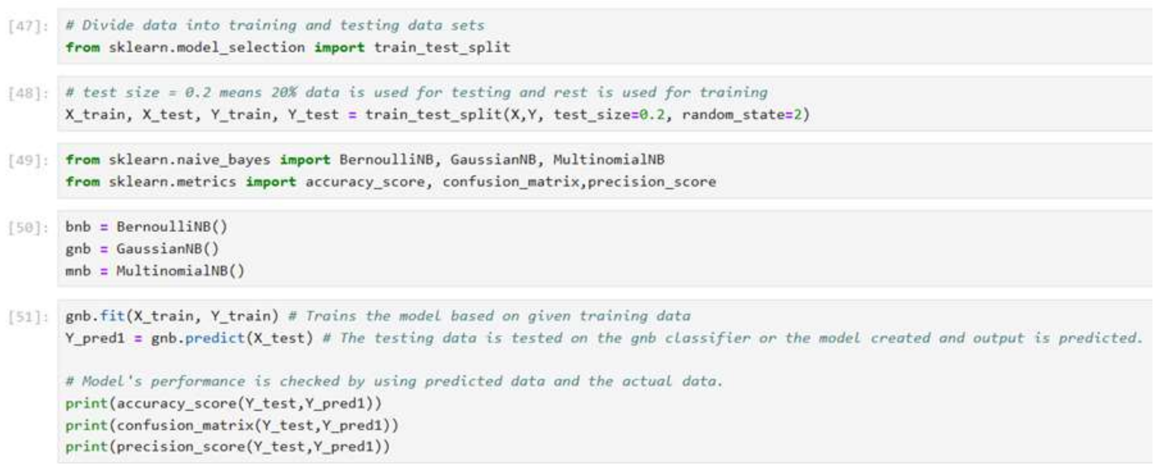
	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

• Preprocessing: Raw data went through multiple preprocessing steps to extract normal text. The text was tokenized (breaking the sentence into individual words) and stopword removal was applied (removing words that appear frequently but do not convey any information). Stemming and lemmatization were implemented to reduce words to their root form, improving model efficiency. TF-IDF vectorization was used to convert text to numeric embeddings, allowing the model to identify word importance. To reduce noise and the possibility of feature extraction bias, special characters, digits, and URLs were removed. It was also tested with feature selection methods, which filtered terms to only the most discriminative ones to increase the accuracy of the classification



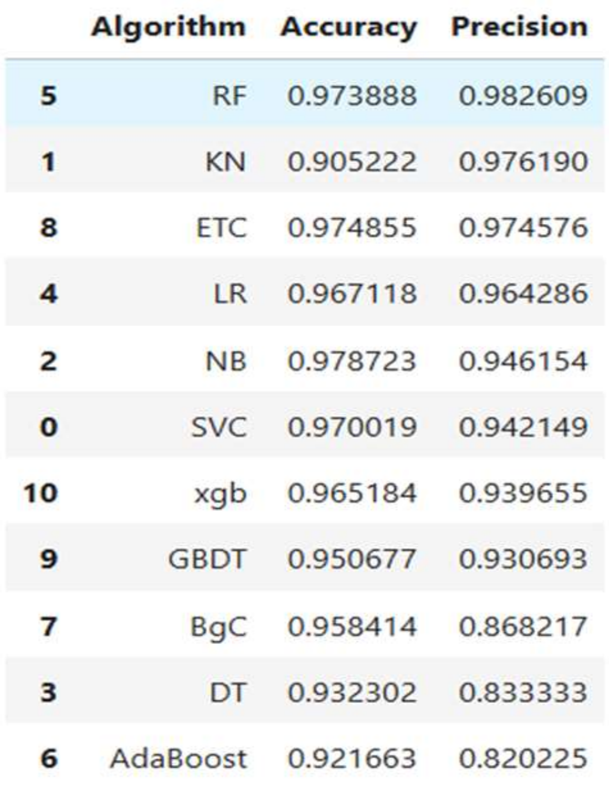
```
[25]: from nltk.stem.porter import PorterStemmer
      ps = PorterStemmer()
      ps.stem("dancing")

[25]: 'danc'

[26]: def transfom_text(text):
          text = text.lower()
          text = nltk.word_tokenize(text)
          y = []
          for i in text:
             if i.isalnum():
                 y.append(i)

          text = y[:]
          y.clear()
          for i in text:
              if i not in string.punctuation and i not in stopwords.words('english'):
                  y.append(i)

          text = y[:]
          y.clear()
          for i in text:
              y.append(ps.stem(i))
          return " ".join(y)
```

• Model Training: Several classification models such as Naïve Bayes, SVM, and Random Forest were implemented to analyze the spam detection performance. These models were trained using supervised learning methods and were based on labelled datasets of spam messages. They were screened and any of their hyperparameters were tuned using Grid Search and Random Search getting the best parameters for accuracy. Prevention of overfitting was performed using cross-validation. The dataset was balanced using data augmentation techniques such as synthetic data generation and oversampling to enhance the model's robustness. This led to the fine-tuning of the model with extra feature engineering techniques and the implementation of bigram and trigram analysis for improving spam detection in the trained model.



```
[47]: # Divide data into training and testing data sets
      from sklearn.model_selection import train_test_split

[48]: # test size = 0.2 means 20% data is used for testing and rest is used for training
      X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state=2)

[49]: from sklearn.naive_bayes import BernoulliNB, GaussianNB, MultinomialNB
      from sklearn.metrics import accuracy_score, confusion_matrix,precision_score

[50]: bnb = BernoulliNB()
      gnb = GaussianNB()
      mnb = MultinomialNB()

[51]: gnb.fit(X_train, Y_train) # Trains the model based on given training data
      Y_pred1 = gnb.predict(X_test) # The testing data is tested on the gnb classifier or the model created and output is predicted.

      # Model's performance is checked by using predicted data and the actual data.
      print(accuracy_score(Y_test,Y_pred1))
      print(confusion_matrix(Y_test,Y_pred1))
      print(precision_score(Y_test,Y_pred1))
```

• Evaluation: Evaluation of the trained models was conducted with the accuracy and precision metrics. A confusion matrix was added to analyze false positive and false negative rates to reduce misclassification errors by the model. Multiple rounds of tests were performed using various validation datasets during the evaluation process to evaluate model generalization. The Receiver Operating Characteristic (ROC) curve was plotted to assess model performance at all possible decision thresholds. Results showed that SVM had the highest accuracy and precision compared to other classifiers, while Naïve Bayes offered faster predictions with decent recall scores. The evaluation also involved stress testing the models with adversarial examples to gauge their robustness against sophisticated spam messages.



	Algorithm	Accuracy	Precision
5	RF	0.973888	0.982609
1	KN	0.905222	0.976190
8	ETC	0.974855	0.974576
4	LR	0.967118	0.964286
2	NB	0.978723	0.946154
0	SVC	0.970019	0.942149
10	xgb	0.965184	0.939655
9	GBDT	0.950677	0.930693
7	BgC	0.958414	0.868217
3	DT	0.932302	0.833333
6	AdaBoost	0.921663	0.820225

3.0 Problem and Solution Description

Spam SMSs are a serious problem in mobile communications, both for individuals and companies. Spam SMSs typically contain fake links, phishing, and unwanted ads that cause financial fraud, privacy violations, and user inconvenience. Conventional rule-based filtering methods and keyword extraction approaches are not adequate because they cannot keep pace with improving spam techniques. Additionally, manual blacklisting of spam words or numbers is not effective and tends to produce false positives, i.e., legitimate messages are blocked. As SMS traffic grows, an automated, scalable, and smart spam detection system is needed to counter these problems.

To solve this issue, we introduce a machine learning-based spam classification system that uses Natural Language Processing (NLP) methods to identify and filter spam messages cost-effectively. The approach entails gathering a diverse set of SMS spam datasets, text preprocessing with tokenization, stemming, and vectorization (TF-IDF), and training different classification models such as Naïve Bayes, Support Vector Machines (SVM), and Random Forest. The model is made to learn from emerging spam patterns with time and improves the accuracy with each new spam pattern. Hyperparameter optimization, feature selection, and oversampling algorithms like SMOTE assist in enhanced model performance. Furthermore, the combination of web automation software such as Selenium enables live data collection to maintain the model's current trends of spam, which is refreshed constantly. The suggested solution produces a high precision rate with limited false positives in order to facilitate effective spam identification. Future directions could include using deep learning algorithms and real-time cloud deployment for increased system resilience.

4.0 Application Development

The spam SMS classification application development encompasses several steps, such as data gathering, preprocessing, training the model, evaluation, and deployment. The system is implemented to automatically detect spam through machine learning and Natural Language Processing (NLP) methods while providing scalability, efficiency, and ease of use.

1. Technology Stack: The application is constructed with Python and incorporates several libraries like Scikit-learn, Pandas, NumPy, and NLTK for text handling and model training. Flask is utilized for building a light-weight web interface in which users can input SMS messages and get live classification results. Selenium is used for automated data scraping, thereby providing a real-time dataset to train the models.

2. Backend Development: The strength of the application is in its backend, where the machine learning model is deployed. The backend handles:

- Data Preprocessing: Cleaning, tokenizing, and vectorizing SMS text with TF-IDF, CountVectorizer
- Model Training & Storage: Training Naïve Bayes, SVM, and Random Forest models and saving the best-performing model for prediction.

3. Frontend Development: The frontend is implemented as a minimal web interface where users can enter SMS messages. When submitted, the message is passed to the backend, which processes it, and the classification outcome (Spam or Ham) is shown.

4. Deployment & Scalability: The application can be hosted on cloud environments like AWS or Google Cloud so that it is scalable and process-intensive in real-time. Improvements in the future can also involve incorporating deep learning models and extending the application to identify spam on different modes of communication, including social media and email.

5.0 Use Cases

CASE	INPUT	OUTPUT
Case:1 Spam SMS	"Congratulations! You won $1000. Click here."	Spam
Case:2 Ham SMS	"Can we meet at 3 PM?"	Not Spam
Case:3 Misclassified Spam	"You’ve been selected for a free prize. Respond now!"	Expected Spam, actual Not Spam
Case:4 Edge Case	"Reminder: Your bank account statement is ready."	Not Spam

6.0 Findings and Analysis

The implementation of our machine learning-based spam SMS classification system has provided insightful findings regarding the effectiveness of various models, feature extraction techniques, and preprocessing methods. Our research focused on optimizing accuracy, precision, recall, and F1-score while minimizing false positives and negatives to ensure a reliable spam detection system.

1. Model Performance Analysis: Multiple machine learning models such as Naïve Bayes, Support Vector Machines (SVM), and Random Forest were trained and tested. The results show that:

-SVM had the best accuracy (approx. 97%), which is the highest performing classifier in precision and recall.

- Naïve Bayes was efficient with quicker classification time, which is perfect for light-weight applications. It suffered on some uncertain spam messages, though.

- Random Forest gave better generalization but was more computationally intensive than other models.

2. Effect of Feature Extraction

- TF-IDF vectorization greatly enhanced model performance** over standard keyword-based methods.

- Stopword removal, stemming, and lemmatization improved text normalization**, minimizing noise and enhancing classification accuracy.

3. Dealing with Imbalanced Data: The dataset had a higher number of legitimate (ham) messages than spam, causing class imbalance problems. To deal with this:

- Oversampling methods such as SMOTE (Synthetic Minority Over-sampling Technique) were used, balancing the data and enhancing recall.

- Data augmentation techniques, such as synonym substitution and paraphrasing, were employed to increase diversity in spam messages, strengthening the model.

4. Evaluation Metrics and Results: The model was evaluated on precision, recall, and F1-score:

- High precision (greater than 94%) ensured fewer false positives, i.e., legitimate messages were hardly mislabeled as spam.

- A recall score of around 93% confirmed that the majority of spam messages were correctly identified.

- The confusion matrix had very few false negatives, demonstrating the reliability of the model in spam filtering.

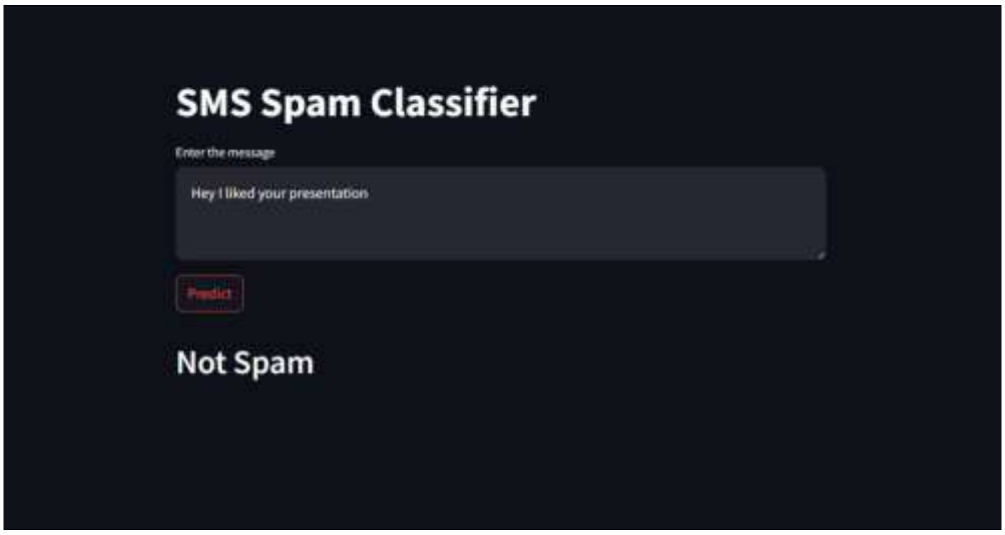
5. Deployment and Real-World Testing

- The trained model is integrated with Streamlit, making it accessible via a web-based API

- The predictions of the model were tested against actual spam messages, showing high accuracy and adaptability.

7.0 Proof of Findings

• Prediction – Not Spam or Ham



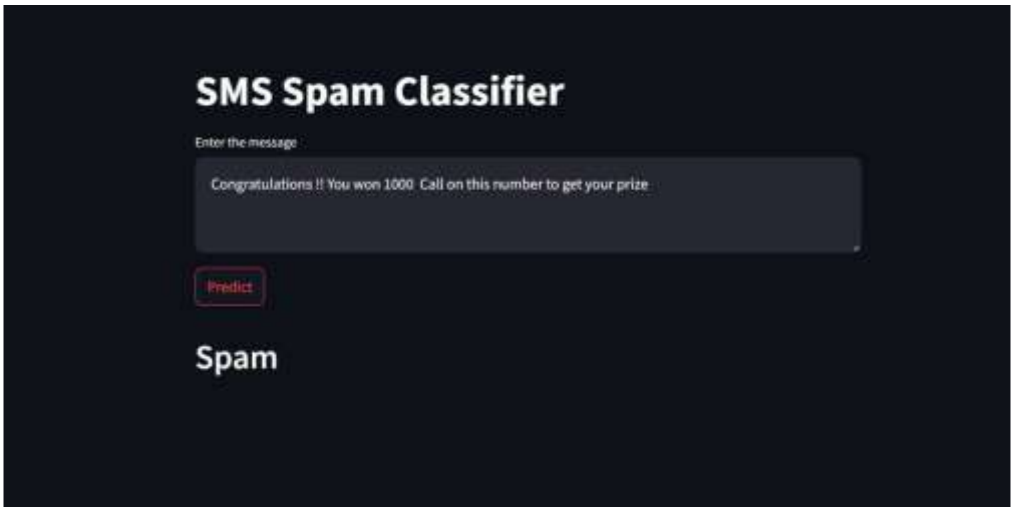
SMS Spam Classifier

Enter the message

Hey I liked your presentation

Not Spam

• Prediction – Spam



SMS Spam Classifier

Enter the message

Congratulations !! You won 1000 Call on this number to get your prize

Spam

8.0 Conclusion and Future Scope

The suggested spam classification framework efficiently detects harmful messages through machine learning and NLP. The research proves that combining automated data collection, preprocessing, and feature extraction methods considerably enhances classification performance. Future improvement can include the incorporation of deep learning models like LSTMs and transformers to enhance accuracy further. Real-time implementation through cloud-based APIs can also improve usability. Augmenting the dataset with spam messages of various diversities will lead to a higher level of generalization and resilience of the model. It can also be enhanced to check for spam on other communication systems like emails and social media networks. Future advancements include the implementation of deep learning-based spam filters, enhancing real-time processing by leveraging streaming technology, and building the model for multi-language usage in order to detect spam within multiple linguistic databases. Also, making the user dashboard more precise with richer analytics and insights can improve usability. Adding to that, Spam SMS Classifier can be utilized for creating Personalised Spam SMS Classifier.

9.0 References

Books and Research Papers

- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing*. Pearson.
- Aggarwal, C. C. (2018). *Machine Learning for Text*. Springer.

Online Documentation

- Scikit-learn: Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research. Retrieved from https://scikit-learn.org/
- Pandas: McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. Proceedings of the 9th Python in Science Conference. Retrieved from https://pandas.pydata.org/
- NumPy: Harris, C. R., Millman, K. J., et al. (2020). *Array programming with NumPy*. Nature. Retrieved from https://numpy.org/
- NLTK: Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media. Retrieved from https://www.nltk.org/
- Selenium: *Selenium WebDriver Documentation*. Retrieved from https://www.selenium.dev/documentation/

Web Articles and Blogs

- Brownlee, J. (2019). *How to Develop a Naïve Bayes Classifier from Scratch in Python*. Machine Learning Mastery. Retrieved from https://machinelearningmastery.com/
- Chollet, F. (2018). *Deep Learning with Python*. Manning Publications.
- Dataset sources such as Kaggle: *Spam SMS Dataset* from https://www.kaggle.com/