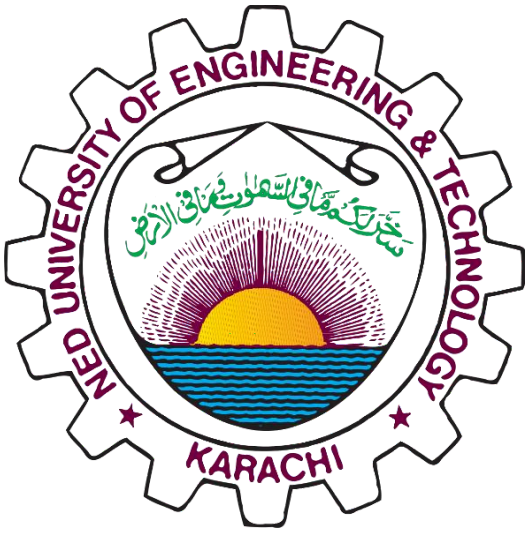
MACHINE LEARNING
(CS-324)
OPEN ENDED LAB



Spam Scanner : Enhanced SMS Spam Detection Using Machine Learning Algorithms Report

SUBMITTTED BY:

YASHAL ALI FAROOQUI (CS-20184)

ALISHBA RIZWAN (CS-21044)

NARMEEN SHAHBAZ (CS-21047)

SUBMITTTED TO:

MS MAHNOOR MALIK

Spam Scanner :Enhanced SMS Spam Detection Using Machine Learning Algorithms Project Report

Introduction

The proliferation of SMS (Short Message Service) as a primary means of communication has led to an increase in spam messages, which can be both intrusive and harmful. This project focuses on developing a robust machine learning model to detect spam messages in SMS. The goal is to build a classifier that accurately differentiates between spam and legitimate messages, leveraging various algorithms and techniques in natural language processing (NLP) and machine learning.

Libraries and Tools Used

- **Pandas:** For data manipulation and analysis.
- **NumPy :** For numerical computations.
- **Scikit-learn:** For implementing various machine learning algorithms and model evaluation.
- **Matplotlib & Seaborn :** For data visualization.
- **NLTK :** For natural language processing tasks.
- **Pickle:** For saving the trained model and vectorizer.
- **Streamlit**: Web framework for creating interactive web applications.

Data Preprocessing

1. **Loading Data**: The dataset was loaded and inspected to understand its structure and contents.
2. **Cleaning Data** : Text data was cleaned by removing special characters, stopwords, and stemming.
3. **Feature Extraction**: TF-IDF (Term Frequency-Inverse Document Frequency) vectorization was used to convert the text data into numerical features suitable for machine learning models.

Model Building and Evaluation

Multiple algorithms were implemented and evaluated to determine the best performing model:

1. K-Nearest Neighbors (KNN)
2. Naive Bayes (NB)
    - Gaussian Naive Bayes (GNB)
    - Multinomial Naive Bayes (MNB)
    - Bernoulli Naive Bayes (BNB)
3. Random Forest (RF)
4. Support Vector Classifier (SVC)
5. Logistic Regression (LR)
6. Decision Tree (DT)

Evaluation Metrics

The models were evaluated using the following metrics:

- **Accuracy**: the ratio of correctly predicted instances to the total instances.
- **Precision**: the ratio of correctly predicted positive observations to the total predicted positives.

The performance of the models was as follows:

Algorithm	Accuracy	Precision
KNN	0.95	1.000
GNB	0.979	0.954
MNB	0.971	1.000
BNB	0.972	1.000
RF	0.974	0.983
SVC	0.976	0.975
LR	0.956	0.960
DT	0.932	0.833

Multinomial Naive Bayesian (MNB) Model

Given the high accuracy and precision of the Multinomial Naive Bayes model, it was chosen as the primary model for this project.

MNB Performance with Python Package

- ➢ **Accuracy: 0.971**
- ➢ **Precision: 1.0**

MNB Performance without Python Package

- ➢ **Accuracy : 0.971**
- ➢ **Precision : 0.98**

This comparison demonstrates that while the manually implemented MNB achieved similar accuracy, the precision was slightly lower than the library implementation. This highlights the benefits of using well-optimized libraries for real-world applications.

Ensemble Methods

To further improve the performance, ensemble methods such as Voting Classifier and Stacking classifiers were employed. The Voting Classifier combined multiple models to make predictions based on the majority vote, while the Stacking Classifier used a meta-classifier to combine the base models.

Final Model Performance

- ➢ **Voting Classifier:**

- Accuracy: 0.977
- Precision: 0.975

➢ **Stacking Classifier:**

- Accuracy: 0.977
- Precision: 0.975

Comparison of Manual and Library Implementations

Comparing the performance of manually implemented algorithms and those using libraries provides valuable insights:

1. Multinomial Naive Bayesian

➢ **With Python Package :**

- Accuracy: 0.971
- Precision: 1.0

➢ **Without Python Package:**

- Accuracy: 0.971
- Precision: 0.98

2. Bernoulli Naive Bayesian

➢ **With Python Package:**

- Accuracy: 0.972
- Precision: 1.0

➢ **Without Python Package :**

- - Accuracy: 0.978
- - Precision: 0.98

3. Logistic Regression

➢ **With Python Package:**

- Accuracy: 0.956

➢ **Without Python Package:**

- Accuracy: 0.867

The accuracy and precision of the manually implemented algorithms are generally lower compared to the library implementations. This difference can be attributed to the extensive optimizations and efficiencies built into the libraries.

Future Expansion

1. **Larger Dataset** : Expanding the dataset to include more diverse examples of spam and ham messages.
2. **Real-time Spam Detection** : Deploying the model as a real-time service that can automatically filter spam messages in messaging applications.
3. **Multi-lingual Support**: Extending the model to support spam detection in multiple languages.

Conclusion

This project successfully developed a spam detection model using various machine learning algorithms. The ensemble methods provided the best performance, with high accuracy and precision. The comparison between manual and library implementations of algorithms highlighted the efficiency and reliability of using well-optimized libraries for real-world applications. Future work will focus on enhancing the model's capabilities and deploying it in real-world applications to mitigate the impact of spam messages on users.

References

- Scikit-learn documentation: https://scikit-learn.org/
- NLTK documentation: https://www.nltk.org/
- Lab manual and notebooks used in lab demonstration

Appendix

- Code Repository: https://github.com/yashal-ali/SpamScanner
- Model and Vectorizer Files: The trained model and TF-IDF vectorizer are saved as'model.pkl` and 'vectorizer.pkl', respectively.

By leveraging machine learning and NLP, this project demonstrates an effective approach to tackling the problem of SMS spam, providing a foundation for future advancements and real-world applications.

The Project's Interface