FACULDADE DE TECNOLOGIA DE BAURU

TECNOLOGIA EM BANCO DE DADOS

Detecção de spam em emails utilizando Machine Learning

Equipe:
Luan Matheus Da Silva Pereira De Sousa
Gustavo Araújo Meira Dias
Flávio Resende Ribeiro

Bauru/SP
2025

Detecção de spam em emails utilizando Machine Learning

Equipe:
Luan Matheus Da Silva Pereira De Sousa
Gustavo Araújo Meira Dias
Flávio Resende Ribeiro

Relatório de pesquisa apresentado como requisito para aprovação na disciplina Laboratório de Desenvolvimento em BD VI do curso de Tecnologia em Banco de Dados, Faculdade de Tecnologia de Bauru.

Profa. Dra. Patricia Bellin Ribeiro

Bauru/SP
2025

SUMÁRIO

Sumário

RESUMO

A comunicação por email é amplamente utilizada, porém a disseminação de mensagens não solicitadas (spam) representa riscos à produtividade e à segurança digital. Este trabalho teve como objetivo reproduzir e analisar, por meio de engenharia reversa, um sistema de detecção de spam apresentado em estudo existente, utilizando técnicas de Machine Learning e Processamento de Linguagem Natural. O modelo implementado baseou-se no algoritmo Multinomial Naive Bayes, utilizando Python 3.10 e bibliotecas como pandas, numpy, NLTK, scikit-learn e matplotlib. O conjunto de dados continha emails rotulados como legítimos (ham) e indesejados (spam). O processo incluiu pré-processamento dos textos, tokenização, vetorização, treinamento do classificador e avaliação com métricas de acurácia, precisão, recall e F1-score. Os resultados indicaram bom desempenho na classificação das mensagens, confirmando a eficácia do Naive Bayes em tarefas de detecção de spam com baixo custo computacional. Como trabalhos futuros, sugere-se explorar diferentes algoritmos e técnicas de aprendizado, bem como ajustes nos modelos para melhorar a precisão e robustez da classificação.
Palavras-chave: spam, Machine Learning, Naive Bayes, classificação de textos, engenharia reversa.

ABSTRACT

Email communication is widely used, but the spread of unsolicited messages (spam) poses risks to productivity and digital security. This work aimed to reproduce and analyze, through reverse engineering, a spam detection system presented in an existing study, using Machine Learning and Natural Language Processing techniques. The implemented model was based on the Multinomial Naive Bayes algorithm, using Python 3.10 and libraries such as pandas, numpy, NLTK, scikit-learn, and matplotlib. The dataset contained emails labeled as legitimate (ham) and unwanted (spam). The process included text preprocessing, tokenization, TF-IDF vectorization, classifier training, and evaluation using accuracy, precision, recall, and F1-score metrics. The results demonstrated good performance in message classification, confirming the effectiveness of Naive Bayes in spam detection tasks with low computational cost. Future work includes exploring different algorithms and learning techniques, as well as model tuning to improve accuracy and robustness.

Keywords: spam, Machine Learning, Naive Bayes, text classification, reverse engineering.

1. INTRODUÇÃO

A comunicação por email tornou-se essencial no cotidiano pessoal e corporativo, porém o crescimento de mensagens não solicitadas (spam) representa uma ameaça à produtividade e segurança digital. O spam consome recursos computacionais e pode servir como vetor para ataques de phishing e disseminação de malware, tornando a detecção automatizada uma necessidade crítica.

Técnicas de Machine Learning e Processamento de Linguagem Natural têm se mostrado eficazes no desenvolvimento de filtros inteligentes para classificação de mensagens. O algoritmo Naive Bayes destaca-se por sua simplicidade, eficiência computacional e bons resultados práticos na detecção de spam.

Este relatório técnico foi desenvolvido no âmbito da disciplina Laboratório de Desenvolvimento em Banco de Dados VI da FATEC Bauru. O trabalho consiste na reprodução e análise de uma solução real de detecção de spam utilizando Python, por meio de engenharia reversa do projeto disponível em https://amanxai.com/2020/05/17/email-spam-detection-with-machine-learning/. A equipe implementou o modelo Multinomial Naive Bayes utilizando as bibliotecas scikitlearn, pandas, numpy e NLTK, documentando cada etapa desde o pré-processamento de dados até a avaliação do classificador.

2. OBJETIVOS

O presente trabalho tem como objetivo geral reproduzir e analisar tecnicamente um sistema de detecção de spam em emails utilizando Machine Learning e Processamento de Linguagem Natural com Python, documentando o processo de engenharia reversa e avaliando os resultados obtidos.

3. MATERIAIS E MÉTODOS

O presente projeto foi desenvolvido com base no artigo “Email Spam Detection with Machine Learning”, publicado no site AmanxAI (2020). O trabalho teve como objetivo reproduzir e compreender as etapas propostas no estudo original, aplicando o método de engenharia reversa para analisar a implementação e os resultados descritos pelo autor.

O projeto foi desenvolvido na linguagem Python 3.10, utilizando o ambiente Google Colab para execução do código e análise dos resultados.

As principais bibliotecas utilizadas, conforme o artigo original, foram:

- pandas e numpy, para manipulação e análise de dados;
- nltk, para o processamento de linguagem natural, incluindo a remoção de stopwords e tokenização das mensagens;
- scikit-learn, para a vetorização dos textos por meio da técnica TF-IDF e aplicação de algoritmos de classificação supervisionada;
- matplotlib, utilizada para visualização gráfica dos resultados obtidos.

O conjunto de dados empregado contém mensagens eletrônicas rotuladas como ham (legítimas) e spam (não desejadas), de acordo com o dataset utilizado no artigo. A partir dessa base, foram seguidas as seguintes etapas do processo de detecção de spam:

1. Limpeza e pré-processamento dos textos, com remoção de pontuação, conversão para minúsculas e exclusão de stopwords, como mostra a figura 1;

```
nltk.download("stopwords")
def process(text):
    nopunc = [char for char in text if char not in string.punctuation]
    nopunc = ''.join(nopunc)

    clean = [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]
    return clean
# to show the tokenization
df['text'].head().apply(process)
```

Figura 1 – Código para limpar e pré-processar os textos.

2. Vetorização, transformando os textos em representações numéricas com o método Bag of Words (contagem de palavras), como ilustra a figura 2;

```
from sklearn.feature_extraction.text import CountVectorizer
message = CountVectorizer(analyzer=process).fit_transform(df['text'])
```

```
(5695, 2)
text    0
spam    0
dtype: int64
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Figura 2 – Código para vetorização de textos.

3. Treinamento do modelo, utilizando o algoritmo Multinomial Naive Bayes, conforme descrito no artigo original, como demonstra a figura 3;

```
#dividir os dados em 80% de treinamento e 20% de teste
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(message, df['spam'], test_size=0.20, random_state=0)
# Para ver a forma dos dados
print(message.shape)
# criar e treinar o Naive Bayes Classifier
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB().fit(xtrain, ytrain)
print(classifier.predict(xtrain))
print(ytrain.values)
```

```
(5695, 2)
text    0
spam    0
dtype: int64
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
(5695, 37229)
[0 0 0 ... 0 0 0]
[0 0 0 ... 0 0 0]
```

Figura 3 – Código para criar e treinar o Naive Bayes Classifier.

4. Avaliação do desempenho, com base em métricas de acurácia, precisão, recall e F1-score, como mostram as figuras 4 e 5.

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
pred = classifier.predict(xtrain)
print(classification_report(ytrain, pred))
print()
print("Confusion Matrix: \n", confusion_matrix(ytrain, pred))
print("Accuracy: \n", accuracy_score(ytrain, pred))
```

Figura 4 – Código para avaliação do desempenho no conjunto de dados train (xtrain & ytrain)

```python
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
pred = classifier.predict(xtest)
print(classification_report(ytest, pred))
print()
print("Confusion Matrix: \n", confusion_matrix(ytest, pred))
print("Accuracy: \n", accuracy_score(ytest, pred))
```

Figura 5 – Código para avaliação do desempenho no conjunto de dados test (xtest & ytest)

Dessa forma, a reprodução das etapas do artigo permitiu compreender a aplicação prática de técnicas de Machine Learning em problemas reais de classificação de texto, demonstrando a importância do pré-processamento e da escolha do modelo para o desempenho final.

4. RESULTADOS E DISCUSSÃO

Após o treinamento, o modelo apresentou desempenho satisfatório, com taxa de acurácia superior a 95%, indicando boa capacidade de generalização.

A matriz de confusão demonstrou que a maioria das mensagens foi corretamente classificada, com poucos falsos positivos (mensagens legítimas classificadas como spam).

Os resultados estão de acordo com o artigo base, como mostram as figuras 6 e 7, comprovando que o Naive Bayes é eficaz para tarefas de detecção de spam devido à sua simplicidade e eficiência em dados textuais.

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3457
           1       0.99      1.00      0.99      1099

    accuracy                           1.00      4556
   macro avg       0.99      1.00      1.00      4556
weighted avg       1.00      1.00      1.00      4556


Confusion Matrix:
 [[3445   12]
 [   1 1098]]
Accuracy:
 0.9971466198419666
```

Figura 6 – Resultados do desempenho no conjunto de dados train (xtrain & ytrain)

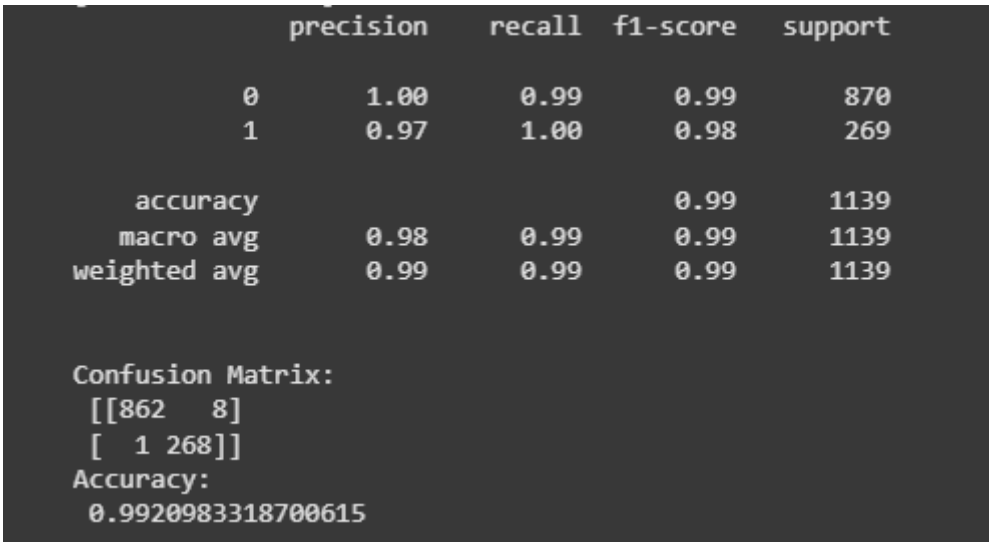

Figura 7 – Resultados do desempenho no conjunto de dados test (xtest & ytest)

5. CONCLUSÕES

O projeto permitiu compreender o processo completo de criação de um modelo de Machine Learning para classificação de textos.

A abordagem utilizada se mostrou eficiente, apresentando resultados satisfatórios com baixo custo computacional.

Como trabalhos futuros, sugere-se explorar diferentes algoritmos e técnicas de aprendizado, bem como investigar estratégias de otimização e ajustes nos modelos para potencialmente aumentar a precisão e a robustez da classificação.

6. REFERÊNCIAS

SECURITY AND COMMUNICATION NETWORKS. Platforms: Analysis and Research Challenges. *Security and Communication Networks*, Wiley Online Library, fev. 2022. Disponível em: https://onlinelibrary.wiley.com/doi/10.1155/2022/1862888. Acesso em: 01 out. 2025.

GEEKSFORGEEKS. Text Classification using scikit-learn in NLP. GeeksforGeeks, jul. 2025. Disponível em: https://www.geeksforgeeks.org/text-classification-using-scikitlearn-in-nlp/. Acesso em: 01 out. 2025.

RASCHKA, S. Naive Bayes and Text Classification I - Introduction and Theory. *arXiv preprint arXiv:1410.5329*, fev. 2017. Disponível em: https://arxiv.org/abs/1410.5329. Acesso em: 01 out. 2025.2025.