




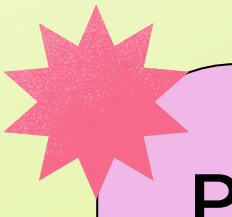
# DETECTOR DE SPAM EM EMAILS

Autores: Alanda Oliveira, Celso Pereira,  
Gustavo Gouveia, Ian Costa, Kléber Araújo  
e Leonardo Benício.

# *FUNCIONALIDADES*



Este projeto implementa um detector de spam em e-mails utilizando um modelo de Naive Bayes e a API do Gmail.



Pré-requisitos: Python 3.x e Bibliotecas Python: scikit-learn, google-auth, google-auth-oauthlib, google-auth-httpplib2, google-api-python-client.

## CONTRIBUIÇÃO:

Diretrizes de estilo de código: Seguir as PEP 8 e PEP 257 para Python.

JUPYTER, VS CODE,  
GOOGLE COLABS.

BACK END FEITO EM  
PYTHON.

# TREINAMENTO

```
def train_spam_detector(texts, labels):
    X_train, X_test, y_train, y_test = train_test_split(texts, labels, test_size=0.2, random_state=42)

    vectorizer = CountVectorizer()
    X_train_vectorized = vectorizer.fit_transform(X_train)
    X_test_vectorized = vectorizer.transform(X_test)

    # Modelo Naive Bayes
    classifier = MultinomialNB()
    classifier.fit(X_train_vectorized, y_train)

    predictions = classifier.predict(X_test_vectorized)

    accuracy = accuracy_score(y_test, predictions)
    report = classification_report(y_test, predictions)

    print(f"Acurácia: {accuracy}")
    print("Relatório de Classificação:")
    print(report)

    return vectorizer, classifier
```

# *Leitura de diferentes corpos de email e opcionalmente anexo.*

```
def list_messages(service, user_id='me', label_ids=['INBOX']):
    result = service.users().messages().list(userId=user_id, labelIds=label_ids).execute()
    messages = result.get('messages', [])
    return messages

def detect_spam_from_email(vectorizer, classifier, service, msg_id):
    message = service.users().messages().get(userId='me', id=msg_id).execute()

    if 'data' in message['payload']['body']:
        email_body = base64.urlsafe_b64decode(message['payload']['body']['data']).decode('utf-8')
    elif 'attachmentId' in message['payload']['body']:
        email_body = "Anexo não processado"
    else:
        email_body = "Corpo de e-mail não encontrado"

    email_vectorized = vectorizer.transform([email_body])
    prediction = classifier.predict(email_vectorized)

    return prediction[0]

texts = ["Este é um email normal.", "Ganhe dinheiro rápido!", "Oferta especial apenas para você."]
labels = [0, 1, 0] # 0 para normal, 1 para spam
```

# *Palavras chave e lógica de retorno*

```
texts = ["Este é um email normal.", "Ganhe dinheiro rápido!", "Oferta especial apenas para você."]
labels = [0, 1, 0] # 0 para normal, 1 para spam

refresh_token = '1//0hcEdrBw_f6tHCgYIARAAGBESNwF-L9IrLXblLDyRqGJpoJpsuhBHYw1HkS-YJokJtlR_surdMx7-P8W9MNCZeFtR9ThnKgneTN0'
credentials_path = '/home/suportetigc/Área de Trabalho/ProjetoML/Credencial.json'

vectorizer, classifier = train_spam_detector(texts, labels)

gmail_service = get_gmail_service(credentials_path, refresh_token)
messages = list_messages(gmail_service)

for msg in messages:
    msg_id = msg['id']
    prediction = detect_spam_from_email(vectorizer, classifier, gmail_service, msg_id)

    print(f"Email ID: {msg_id}")
    print(f"Detecção de Spam: {'Spam' if prediction == 1 else 'Não é Spam'})
```