

Instituto Politécnico do Porto
Escola Superior de Tecnologia e Gestão de Felgueiras

Spam or Non-spam

Licenciatura em Engenharia Informática

Inteligência Artificial

Docente - Davide Carneiro

Discente - Sérgio Félix (8200615)

2023/2024

Índice

Índice de ilustrações	3
Introdução.....	4
1. Dataset & Problema.....	5
2. Processo de Treino.....	6
3. Avaliação dos Modelos	11
4. Interface Gráfica.....	19
5. Conclusão	23

Índice de ilustrações

Figura 1 - Função de transformação dos dados.....	7
Figura 2 - Função de processamento do texto.....	8
Figura 3 - Conversão de texto para vetores numéricos	8
Figura 4 - Processo de aprendizagem de ML.....	9
Figura 5 - Divisão dos dados para treino, validação e teste.....	9
Figura 6 - Modelos escolhidos e parâmetros usados	10
Figura 7 - Resultado das métricas em gráfico de barras	11
Figura 8 - Curva de aprendizagem inicial do modelo Random Forest	13
Figura 9 - Curva de aprendizagem final do modelo Random Forest.....	14
Figura 10 - Curva de aprendizagem inicial do modelo Logistic Regression...	14
Figura 11 - Curva de aprendizagem final do modelo Logistic Regression.....	15
Figura 12 - Curva de aprendizagem inicial do modelo Support Vector Machine	15
Figura 13 - Curva de aprendizagem final do modelo Support Vector Machine	16
Figura 14 - Matriz de confusão do modelo Random Forest	17
Figura 15 - Matriz de confusão do modelo Logistic Regression.....	17
Figura 16 - Matriz de confusão do modelo Support Vector Machine.....	18
Figura 17 - Feature da escolha do modelo a utilizar	19
Figura 18 - Endpoint /predict	21

Introdução

Neste relatório, será abordado o desenvolvimento de um sistema de classificação de mensagens de correio eletrônico utilizando técnicas de aprendizagem automática. O objetivo principal é identificar se uma mensagem é spam ou não, baseado em diversas características presentes no conteúdo das mensagens. Para isso, foi utilizado um dataset específico que contém informações detalhadas sobre cada mensagem, permitindo a aplicação de algoritmos de aprendizagem automática para a classificação. Este relatório está organizado da seguinte forma: na seção 1 é descrito o dataset utilizado e o problema a ser resolvido; na seção 2, detalha-se o processo de treino dos modelos; na seção 3, apresenta-se a avaliação dos modelos desenvolvidos; na seção 4, descreve-se a interface gráfica criada para a visualização e interação com os resultados; e na seção 5, são discutidas as conclusões obtidas.

1. Dataset & Problema

O dataset utilizado neste projeto é composto pelas seguintes colunas:

- Sender: o remetente da mensagem.
- Receiver: o destinatário da mensagem.
- Date: a data em que a mensagem foi enviada.
- Subject: o assunto da mensagem.
- Body: o corpo da mensagem.
- Label: a etiqueta que indica se a mensagem é spam (1) ou não (0).
- Urls: indica se na mensagem continha urls (1) ou não (0).

Relevância no Domínio do Dataset

A identificação de spam é um problema crítico no domínio dos sistemas de correio eletrônico. Mensagens de spam podem ser não só um incômodo, mas também representar uma ameaça à segurança, contendo frequentemente links maliciosos que podem levar a ataques de phishing ou instalação de malware. Assim, a capacidade de identificar e filtrar automaticamente estas mensagens é de extrema importância para garantir a segurança e a eficiência das comunicações por correio eletrônico.

2. Processo de Treino

O processo de treino dos modelos de aprendizagem automática foi conduzido de forma estruturada, abrangendo várias etapas essenciais, desde o pré-processamento dos dados até a avaliação dos modelos. Abaixo está uma descrição detalhada de cada etapa, baseada no código fornecido.

Transformação dos Dados

Primeiramente, os dados foram carregados e transformados para eliminar colunas irrelevantes e limpar os valores. A transformação foi realizada da seguinte forma:

1. Remoção de Colunas Irrelevantes: As colunas receiver, date, subject e urls foram removidas, uma vez que não são necessárias para a identificação de spam.
2. Tratamento de Valores Nulos: As linhas com valores nulos nas colunas body foram removidas.
3. Remoção de Duplicados: As linhas duplicadas foram eliminadas para evitar redundâncias.
4. Extração de Endereços de Email: A coluna sender foi transformada para extrair apenas o endereço de email.
5. Limpeza do Texto: Foram removidas quebras de linha na coluna body.
6. Renomeação das Colunas: As colunas foram renomeadas para email, message e spam para maior clareza.



```
1 def transform_data(dataframe):
2     dataframe.drop(columns=['receiver', 'date', 'subject', 'urls'], inplace=True)
3     dataframe.dropna(subset=['sender', 'body', 'label'], inplace=True)
4     dataframe.drop_duplicates(inplace=True)
5     dataframe['sender'] = dataframe['sender'].apply(
6         lambda text: re.search(r'<[^>+>+', text).group(1) if re.search(r'<[^>+>+', text) else None)
7     dataframe['body'] = dataframe['body'].apply(lambda x: x.replace('\n', ' '))
8     dataframe.rename(columns={"sender": "email", "body": "message", "label": "spam"}, inplace=True)
9     return dataframe
```

Figura 1 - Função de transformação dos dados

Pré-processamento do Texto

Para preparar o texto para a modelagem, foi realizado um pré-processamento que inclui:

1. Tokenização e Limpeza: O texto foi convertido para minúsculas e todos os sinais de pontuação foram removidos.
2. Remoção de Stopwords: Foram removidas palavras comuns e pouco informativas (stopwords).
3. Stemming: As palavras foram reduzidas às suas raízes usando o algoritmo de Porter Stemmer.



```

1 def text_preprocessing(dataframe):
2     stemmer = PorterStemmer()
3     stopwords_set = set(stopwords.words('english'))
4     corpus = []
5     for i in range(len(dataframe)):
6         text = dataframe['message'].iloc[i].lower()
7         text = text.translate(str.maketrans('', '', string.punctuation)).split()
8         text = [stemmer.stem(word) for word in text if word not in stopwords_set]
9         text = ' '.join(text)
10        corpus.append(text)
11    return corpus

```

Figura 2 - Função de processamento do texto

Extração de Características

Foi utilizada a técnica TF-IDF (Term Frequency-Inverse Document Frequency) para converter o texto pré-processado em vetores numéricos, o que facilita a aplicação dos algoritmos de aprendizagem automática.



```

1 vectorizer = TfidfVectorizer(max_features=10000, min_df=5, max_df=0.7)
2 x = vectorizer.fit_transform(corpus)
3 y = transformed_df.spam

```

Figura 3 - Conversão de texto para vetores numéricos

Divisão dos Dados

O dataset foi dividido em conjuntos de treino, validação e teste, utilizando uma proporção de 60% para treino, 20% para validação e 20% para teste. Esta divisão ajuda a garantir que o modelo não se sobreajuste aos dados de treino e permite uma avaliação justa do desempenho do modelo.

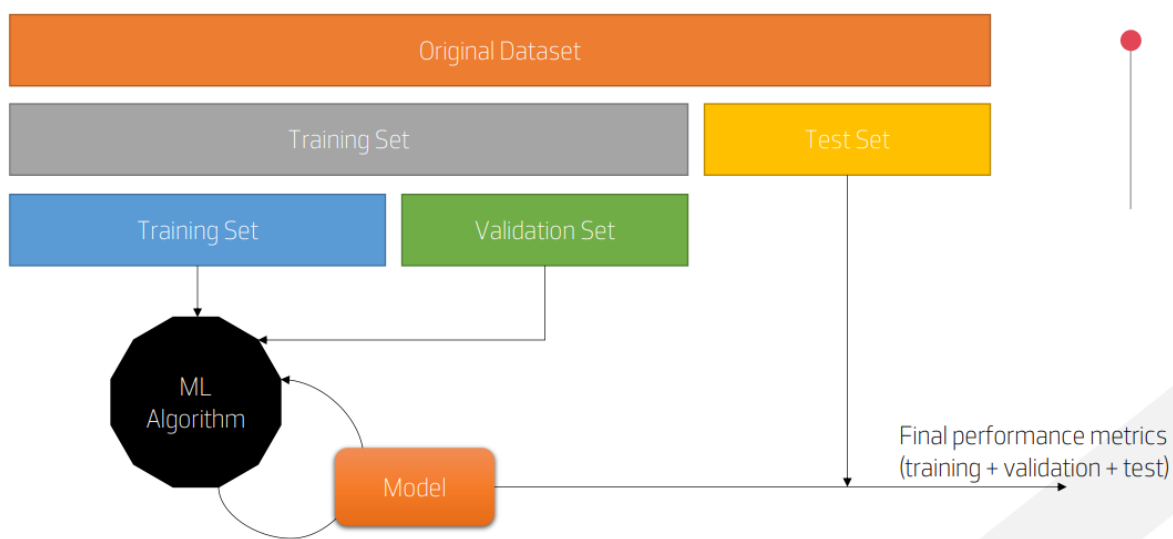


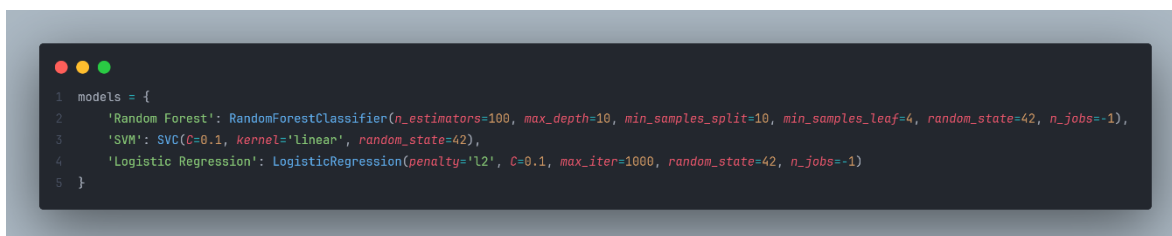
Figura 4 - Processo de aprendizagem de ML

```
1 # Split the data into 60% training, 20% validation, and 20% testing
2 X_train, X_temp, y_train, y_temp = train_test_split(x, y, test_size=0.4, random_state=42)
3 X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
```

Figura 5 - Divisão dos dados para treino, validação e teste

Treino

Foram testados três algoritmos principais: Random Forest, Support Vector Machine (SVM) e Logistic Regression.



```
1 models = {  
2     'Random Forest': RandomForestClassifier(n_estimators=100, max_depth=10, min_samples_split=10, min_samples_leaf=4, random_state=42, n_jobs=-1),  
3     'SVM': SVC(C=0.1, kernel='linear', random_state=42),  
4     'Logistic Regression': LogisticRegression(penalty='l2', C=0.1, max_iter=1000, random_state=42, n_jobs=-1)  
5 }
```

Figura 6 - Modelos escolhidos e parâmetros usados

3. Avaliação dos Modelos

Para avaliar o desempenho dos modelos de classificação binomial treinados, foram utilizadas várias métricas de desempenho, incluindo accuracy, precision, recall e F1 score. Além disso, foram realizadas validações cruzadas para verificar a robustez dos modelos. Abaixo está uma tabela com os resultados das métricas para cada modelo:

Model	Accuracy	Precision	Recall	F1 Score	Cross Validation
<i>Random Forest</i>	0.9393	0.9061	0.995	0.9485	0.9353
<i>Support Vector Machines</i>	0.9862	0.9831	0.9925	0.9878	0.9865
<i>Logistic Regression</i>	0.9811	0.9779	0.9886	0.9833	0.9784

Tabela 1 - Resultado das métricas

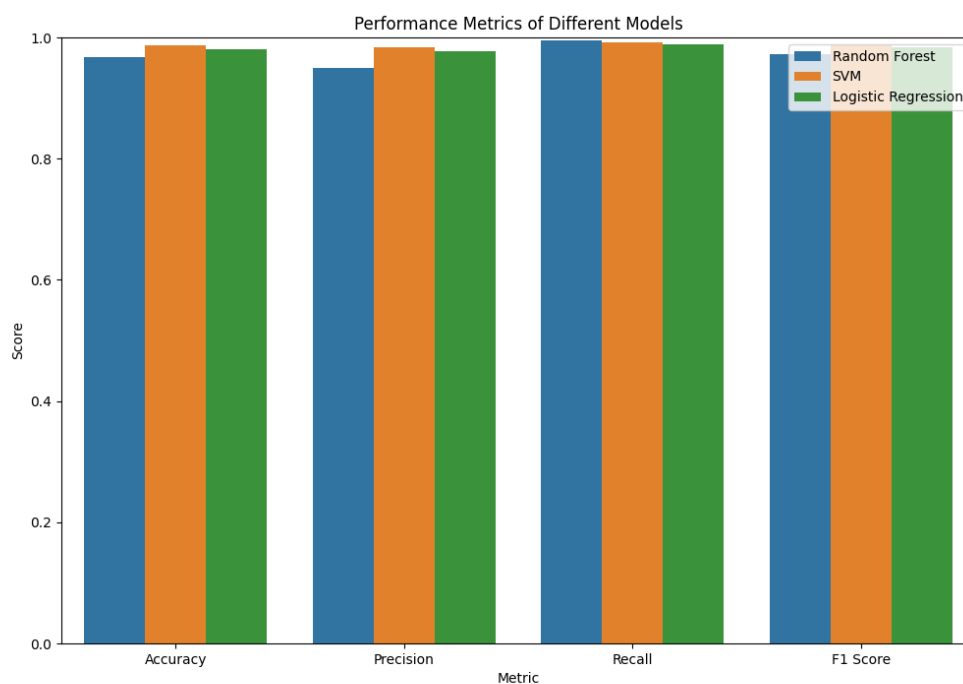


Figura 7 - Resultado das métricas em gráfico de barras

- Accuracy: Mede a proporção total de previsões corretas (tanto verdadeiros positivos quanto verdadeiros negativos) em relação ao total de amostras.
- Precision: Indica a proporção de verdadeiros positivos entre todas as amostras classificadas como positivas.
- Recall: Mede a capacidade do modelo de identificar todas as amostras positivas (spam).
- F1 Score: É a média “absoluta” entre precisão e recall, oferecendo um balanço entre os dois.
- Cross Validation: Média das pontuações de acurácia obtidas através de validação cruzada, que ajuda a avaliar a robustez e a capacidade de generalização do modelo.

O Support Vector Machines (SVM) foi o que apresentou as melhores métricas em todas as categorias, indicando uma alta capacidade de identificar corretamente mensagens spam e não spam, no entanto, é preciso ter em conta que há um grande risco de overfitting, sugerido pela alta precisão e recall.

De seguida o Logistic Regression, desempenho muito próximo ao Support Vector Machine (SVM), com ligeiramente menos precisão e recall, mas também mostra sinais de overfitting, mas é bastante robusto.

Por fim, o Random Forest embora tenha a precisão mais baixa, apresenta um bom equilíbrio entre as métricas, sendo o modelo mais equilibrado sugere que este pode ser mais confiável em diferentes conjuntos de dados, com menor risco de overfitting.

Curvas de Aprendizagem

As curvas de aprendizagem foram geradas para cada modelo para entender o seu comportamento durante o treino. Inicialmente, todos os modelos apresentaram overfitting, especialmente o Support Vector Machine (SVM) e a

Logistic Regression. Após ajustes nos parâmetros, o modelo Random Forest apresentou uma curva de aprendizagem mais equilibrada, enquanto os outros dois modelos ainda mostraram sinais de overfitting.

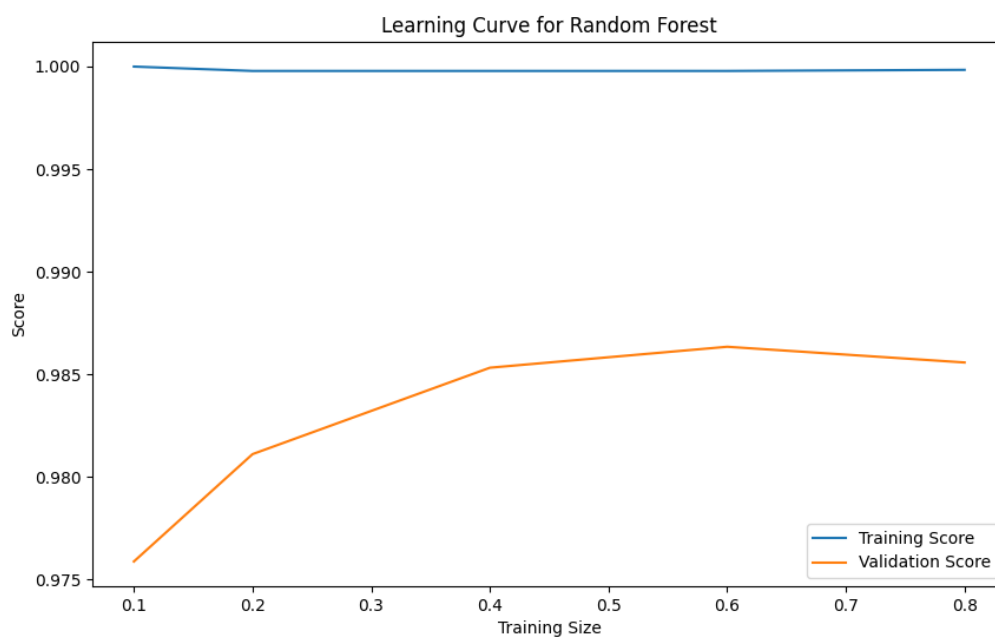


Figura 8 - Curva de aprendizagem inicial do modelo Random Forest

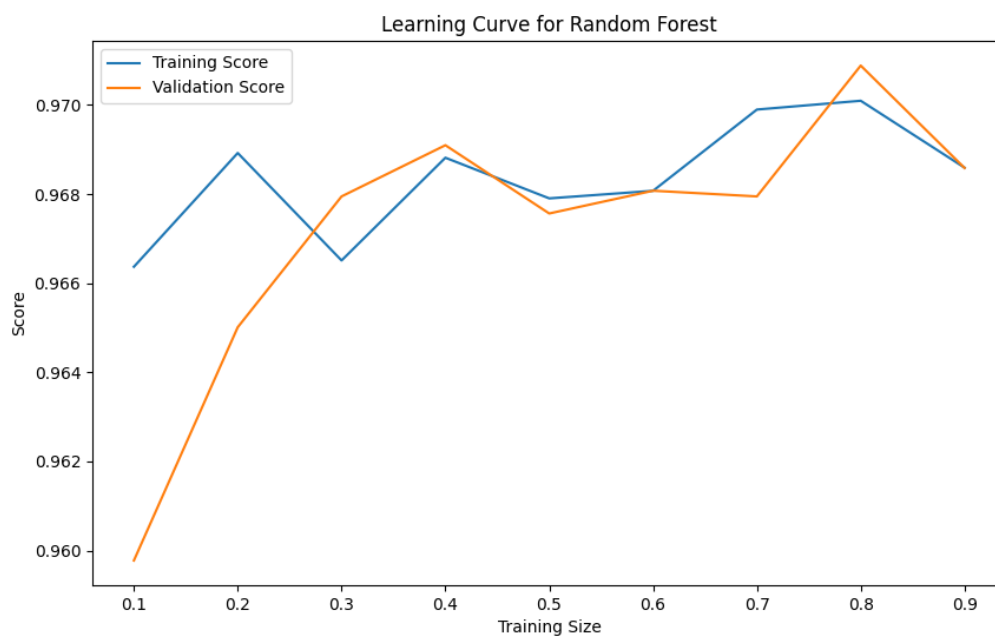


Figura 9 - Curva de aprendizagem final do modelo Random Forest

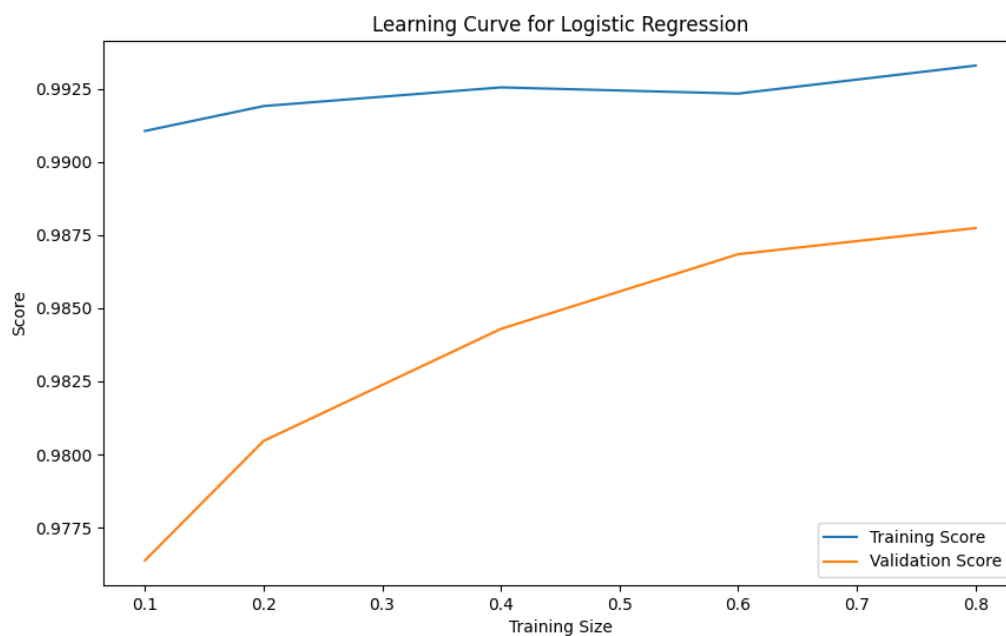


Figura 10 - Curva de aprendizagem inicial do modelo Logistic Regression

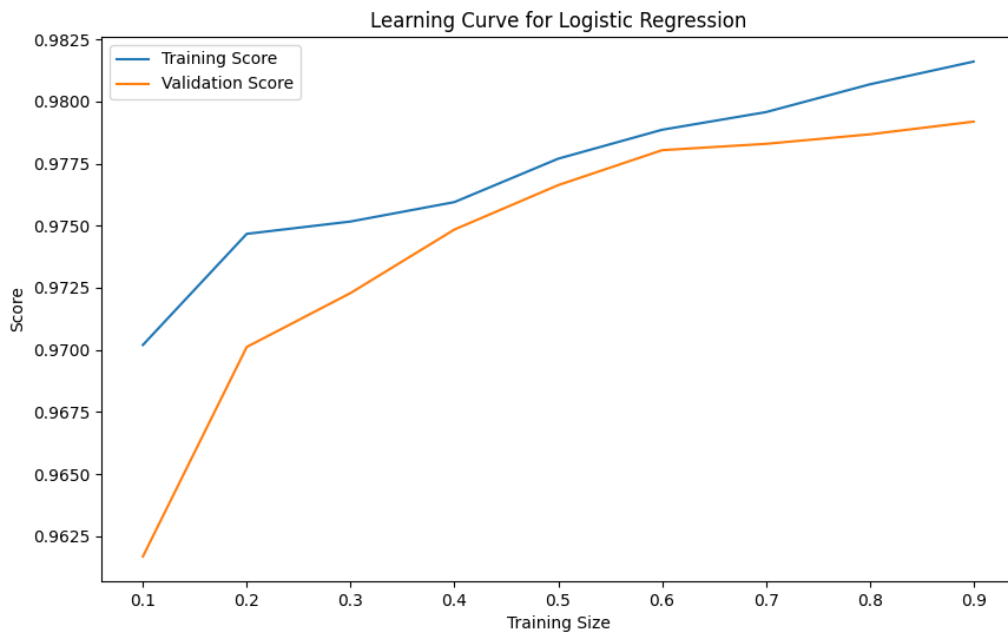


Figura 11 - Curva de aprendizagem final do modelo Logistic Regression

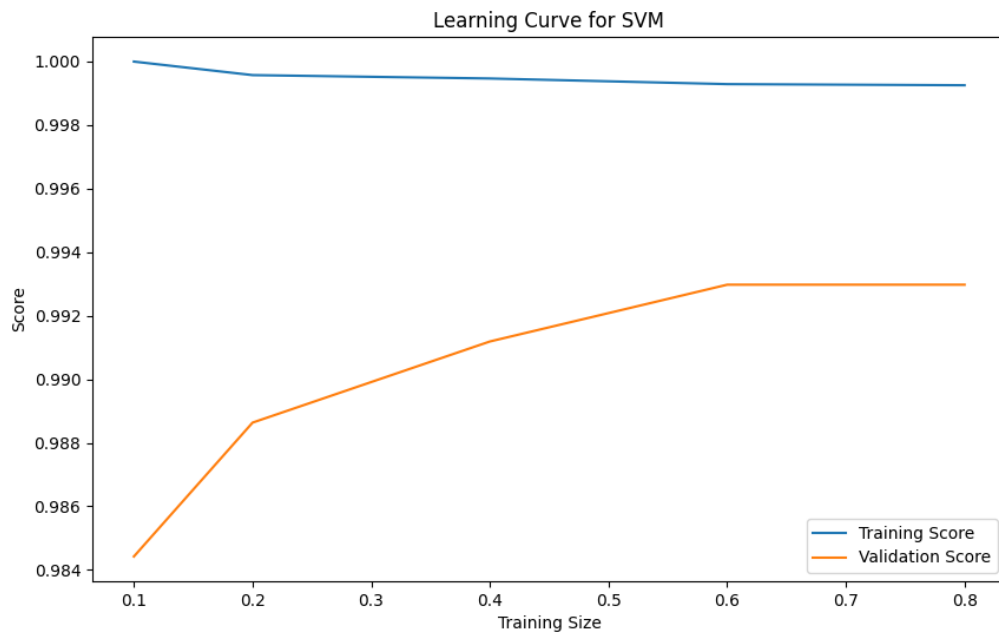


Figura 12 - Curva de aprendizagem inicial do modelo Support Vector Machine

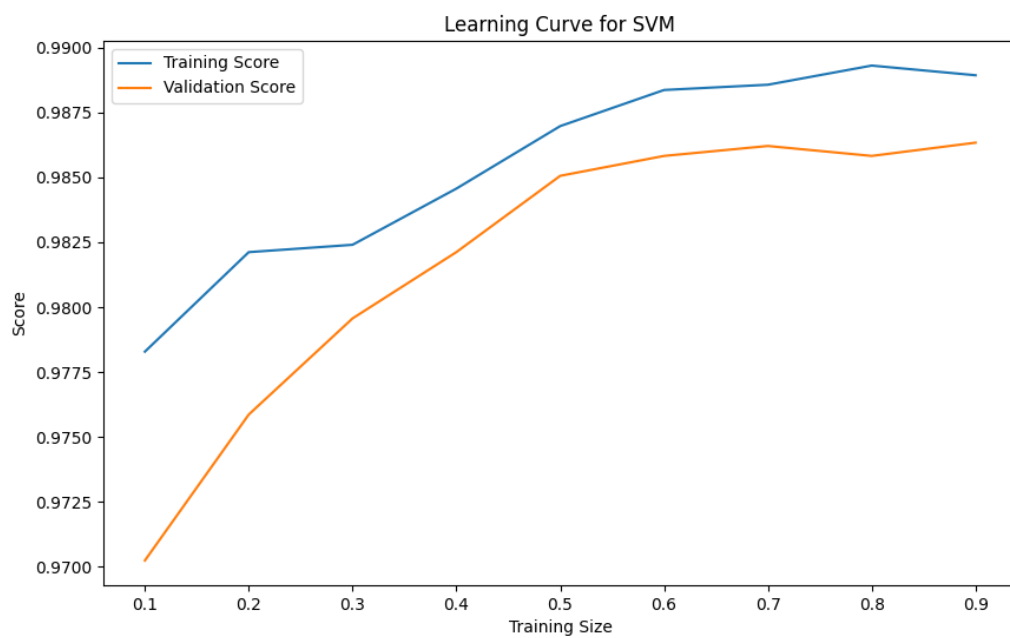


Figura 13 - Curva de aprendizagem final do modelo Support Vector Machine

Matrizes de Confusão

As matrizes de confusão ajudam a entender melhor como os modelos estão classificando as mensagens como spam ou não spam. Abaixo estão as matrizes de confusão para cada modelo:

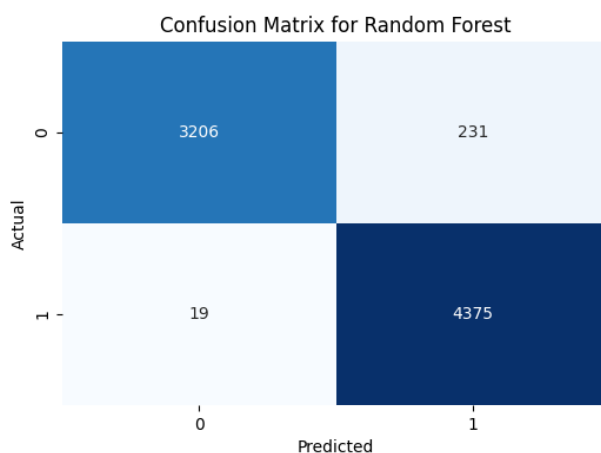


Figura 14 - Matriz de confusão do modelo Random Forest

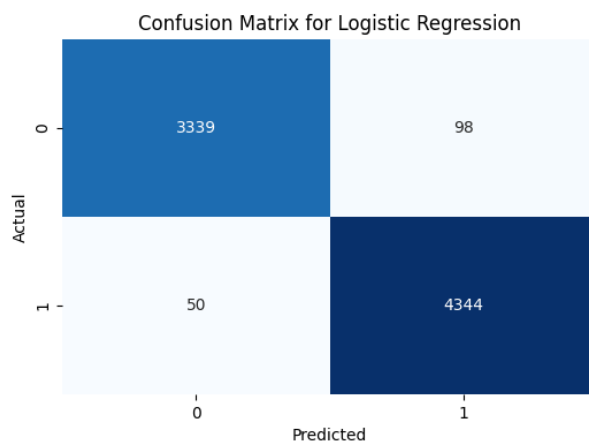


Figura 15 - Matriz de confusão do modelo Logistic Regression

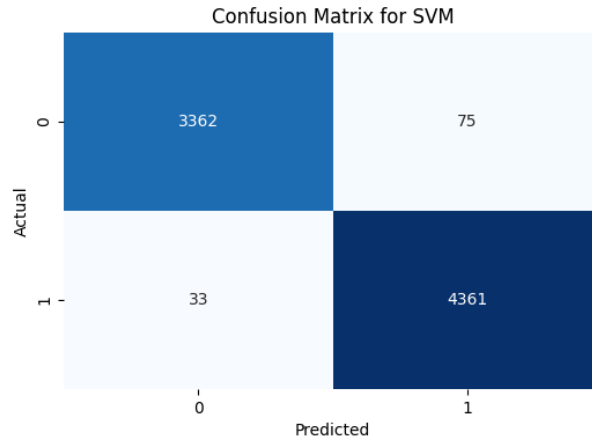


Figura 16 - Matriz de confusão do modelo Support Vector Machine

- Random Forest:
 - False Positives (231): Mensagens não spam classificadas como spam.
 - False Negatives (19): Mensagens spam classificadas como não spam.
 - Este modelo tem uma maior quantidade de falsos positivos em relação aos outros, mas ainda mantém uma alta precisão.
- Support Vector Machine (SVM):
 - False Positives (75): Mensagens não spam classificadas como spam.
 - False Negatives (33): Mensagens spam classificadas como não spam.
 - O SVM apresenta a menor quantidade de falsos positivos, mas ainda sofre de overfitting.
- Logistic Regression:
 - False Positives (98): Mensagens não spam classificadas como spam.
 - False Negatives (50): Mensagens spam classificadas como não spam.

- Embora apresente um bom desempenho, o Logistic Regression também mostra sinais de overfitting.

4. Interface Gráfica

A interface gráfica do projeto foi desenvolvida utilizando a framework Next.js. Esta escolha permitiu criar uma aplicação frontend eficiente, com uma série de funcionalidades úteis para os utilizadores. Entre as principais características da interface, destacam-se:

Escolha do Modelo: O utilizador pode seleccionar um dos três modelos de classificação disponíveis (Random Forest, Support Vector Machine ou Logistic Regression) para fazer as previsões.

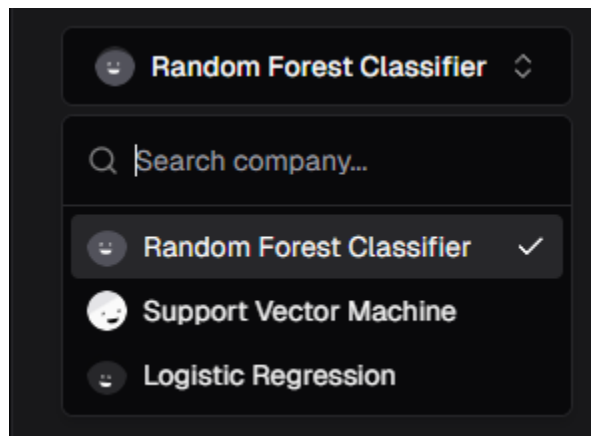
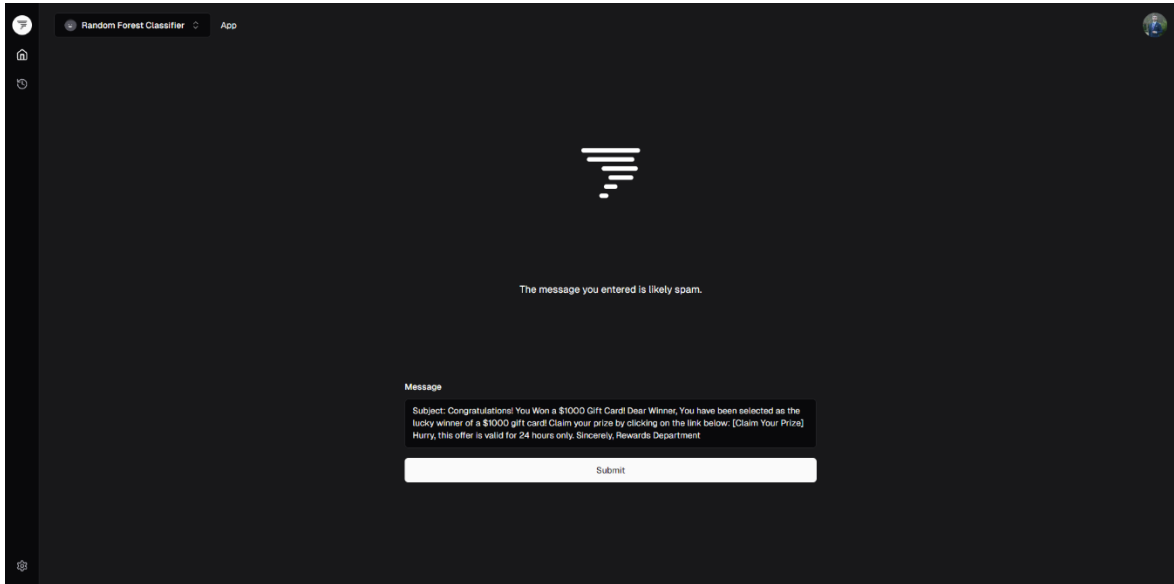
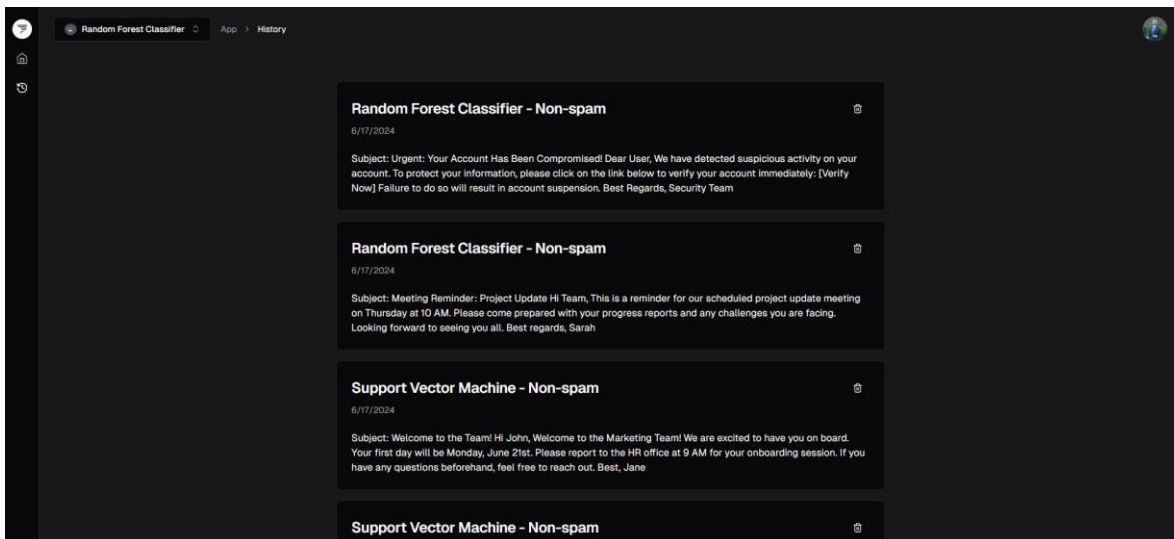


Figura 17 - Feature da escolha do modelo a utilizar

Input da Mensagem: Um campo onde o utilizador pode inserir a mensagem do email que deseja verificar.



Visualização do Histórico: Exibe o histórico dos prompts inseridos pelo utilizador, incluindo o resultado da previsão, a data em que a previsão foi feita e o modelo utilizado, bem como a possibilidade de eliminar o prompt inserido.



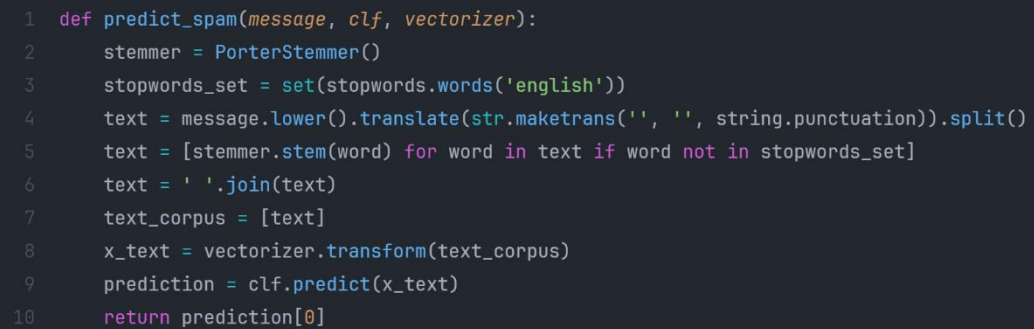
A interface gráfica comunica com o backend através de uma API desenvolvida em Flask. O principal endpoint da API é /predict, que é responsável por receber os dados da mensagem do email e devolver a previsão de spam ou não spam. O funcionamento detalhado é o seguinte:

1. Recepção do Input: O endpoint /predict recebe a mensagem do email enviada pelo frontend.

```
1 @app.route('/predict', methods=['POST'])
2 def predict():
3     try:
4         data = request.get_json(force=True)
5         model = data['model']
6         prompt = data['prompt']
7
8         if not model or not prompt:
9             return jsonify({'error': 'Invalid input'}), 400
10
11         clf, vectorizer = load_models(model)
12
13         prediction = predict_spam(prompt, clf, vectorizer)
14
15         result = True if prediction == 1 else False
16
17         return jsonify({'spam': result})
18     except Exception as e:
19         app.logger.error(f"Error during prediction: {str(e)}")
20         return jsonify({'error': str(e)}), 500
```

Figura 18 - Endpoint /predict

2. Carregamento do Modelo: Dependendo da escolha do utilizador (Random Forest, Support Vector Machine (SVM) ou Logistic Regression), o backend carrega o modelo treinado correspondente.
3. Previsão: O modelo seleccionado processa a mensagem e faz a previsão de se é spam ou não.



```
1 def predict_spam(message, clf, vectorizer):
2     stemmer = PorterStemmer()
3     stopwords_set = set(stopwords.words('english'))
4     text = message.lower().translate(str.maketrans('', '', string.punctuation)).split()
5     text = [stemmer.stem(word) for word in text if word not in stopwords_set]
6     text = ' '.join(text)
7     text_corpus = [text]
8     x_text = vectorizer.transform(text_corpus)
9     prediction = clf.predict(x_text)
10    return prediction[0]
```

4. Resposta: A API retorna o resultado da previsão ao frontend, que então exibe esta informação ao utilizador.

Este sistema integrado garante que o utilizador possa facilmente interagir com os modelos de classificação de spam, inserir novas mensagens para análise, visualizar resultados anteriores e gerir o histórico de previsões de forma eficiente e intuitiva.

5. Conclusão

Neste trabalho prático de Inteligência Artificial, abordámos a classificação de mensagens de email como spam ou não spam utilizando três modelos diferentes: Random Forest, Support Vector Machine (SVM) e Logistic Regression. O dataset foi processado cuidadosamente, removendo colunas irrelevantes e aplicando técnicas de pré-processamento de texto para melhorar a qualidade dos dados de entrada.

Os resultados das avaliações dos modelos mostraram que:

- Support Vector Machine (SVM) apresentou a melhor performance em termos de precisão, recall e F1 score, mas mostrou sinais de overfitting.
- Logistic Regression também teve um desempenho elevado, mas, similar ao Support Vector Machine (SVM), sofre de overfitting.
- Random Forest teve métricas ligeiramente inferiores, mas a sua curva de aprendizagem foi mais equilibrada, indicando um menor risco de overfitting e maior capacidade de generalização.

Para abordar o problema de overfitting nos modelos Support Vector Machine (SVM) e Logistic Regression, podem ser consideradas as seguintes estratégias:

- Aumento do Dataset: Obter mais dados de treino pode ajudar os modelos a generalizar melhor.
- Regularização: Ajustar os parâmetros de regularização (C no caso do Support Vector Machine (SVM) e do Logistic Regression) para penalizar a complexidade excessiva do modelo.
- Características: Adicionar novas características ou utilizar técnicas de seleção de características para manter apenas as mais relevantes.
- Métodos de conjunto: Combinar diferentes modelos para aproveitar as vantagens de cada um e reduzir a variância total.

Em suma, através deste projeto, foi possível entender os desafios e os vários aspectos envolvidos na classificação de spam utilizando diferentes abordagens de machine learning. A integração de um frontend desenvolvido em Next.js e um backend em Flask proporcionou uma interface intuitiva e eficiente para os utilizadores interagirem com os modelos de classificação.

Com base nos resultados obtidos, o modelo Random Forest demonstrou ser uma opção mais equilibrada e robusta, especialmente importante para cenários onde a capacidade de generalização é crucial. No entanto, com as sugestões para mitigar o overfitting, tanto o Support Vector Machine (SVM) quanto o Logistic Regression podem ser melhorados, potencialmente resultando em modelos ainda mais precisos e confiáveis.

Continuar a explorar e implementar essas melhorias pode levar a um sistema de classificação de spam mais eficiente, capaz de lidar com um maior volume de dados e fornecer previsões mais precisas.