Controle Financeiro via WhatsApp

Levy Araújo de Lima
FLD209111CET
Guilherme Augusto Reis dos Santos

RESUMO

Este trabalho apresenta o desenvolvimento de uma plataforma inovadora de controle financeiro pessoal que utiliza o WhatsApp como principal meio de interação com o usuário. A proposta surgiu da percepção da carência por soluções acessíveis, simples e eficazes, especialmente voltadas a usuários com pouca familiaridade com tecnologia. Aproveitando a popularidade e a facilidade de uso do WhatsApp, foi implementada uma interface conversacional por meio de um robô inteligente capaz de compreender comandos em linguagem natural, como “paguei 1200 aluguel” ou “recebi 5200 de salário 06/06”. A partir dessas interações, o sistema registra automaticamente as transações e gera visualizações gráficas dos dados financeiros do usuário, facilitando a análise e a organização do orçamento pessoal. O objetivo central deste trabalho é descrever como técnicas de machine learning foram aplicadas para interpretar a intenção do usuário nas mensagens recebidas. Por meio da aprendizagem de máquina, o sistema é capaz de identificar diferentes tipos de operações financeiras mencionadas em texto livre, contribuindo para uma experiência mais fluida, intuitiva e eficiente no gerenciamento de finanças pessoais. A integração entre uma tecnologia amplamente acessível e modelos de inteligência artificial reflete o potencial transformador das soluções digitais no dia a dia das pessoas.

Palavras-chave: Machine learning; Inteligência artificial; WhatsApp.

Repositório: https://github.com/levyaraujo/caderneta

INTRODUÇÃO

Este projeto propõe o desenvolvimento de uma plataforma de controle financeiro pessoal que utiliza o WhatsApp como principal interface de interação com o usuário. A motivação central para esta iniciativa foi a identificação da necessidade por uma solução

extremamente simples e acessível, capaz de atender até mesmo os usuários com pouca familiaridade com tecnologia. A escolha do WhatsApp se justifica por sua ampla adoção e facilidade de uso. Por trás da interface, há um robô inteligente que interpreta mensagens em linguagem natural, como “paguei 1200 aluguel” ou “recebi 5200 de salário 06/06”, armazena automaticamente os lançamentos e gera visualizações gráficas com base nos dados coletados.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Categorização de texto com machine learning

A categorização de texto é a tarefa de atribuir um valor booleano a cada par ⟨dj, ci⟩ ∈ D × C, onde D é um domínio de documentos e C = {c_1, ..., c|C|} é um conjunto de categorias predefinidas. (Sebastiani, 2001)

O principal objetivo da categorização de textos é a classificação de documentos em um número fixo de categorias predefinidas. Cada documento pode estar em múltiplas, em apenas uma ou em nenhuma categoria. Utilizando aprendizado de máquina, o objetivo principal é ensinar classificadores por meio de exemplos, que realizam automaticamente a atribuição das categorias. Trata-se de um problema de aprendizado supervisionado. Para evitar a sobreposição de categorias, cada categoria é considerada como um problema isolado de classificação binária (Dasari e Rao, 2012).

2.1.1 Processos da classificação de texto

a) Tokenização

Segundo Dasari e Rao (2012), a tokenização é uma etapa fundamental no processamento de linguagem natural (PLN) que consiste em dividir um texto em unidades menores chamadas *tokens*, que podem ser palavras, números, símbolos ou outras estruturas linguísticas relevantes. Esses tokens servem como entrada para as fases subsequentes da classificação de textos.

Em geral, a tokenização ocorre em nível de palavras, utilizando heurísticas simples, como considerar sequências contínuas de letras ou números como um único token, separados por espaços em branco ou sinais de pontuação. Embora essa abordagem funcione bem em idiomas como o inglês, onde as palavras são naturalmente separadas por espaços, ela apresenta limitações em línguas como o chinês, que não possuem delimitadores explícitos entre palavras. Além disso, há dificuldades em tratar expressões compostas, como "Nova York", que idealmente deveriam ser interpretadas como um único token. Para contornar essas limitações, podem ser empregadas técnicas mais avançadas, como heurísticas aprimoradas, consulta a tabelas de expressões frequentes ou

a utilização de modelos de linguagem capazes de identificar e preservar tais colaborações semânticas.

b) Stemming

> Na morfologia linguística e na recuperação de informações, **stemming** é o processo de reduzir palavras derivadas (ou às vezes flexionadas) à sua **raiz** ou **forma original**. A raiz resultante não precisa ser idêntica à raiz morfológica da palavra; geralmente, é suficiente que ela represente um grupo de palavras com **significados semelhantes**, mesmo que essa raiz não seja uma palavra válida por si só. Na ciência da computação, algoritmos de stemming vêm sendo estudados desde 1968. Muitos mecanismos de busca consideram palavras com a mesma raiz como **sinônimos**, em um processo de **ampliação de consulta**, conhecido como **conflation**. (Dasari e Rao, 2012)

c) Stop word removal

A **remoção de stopwords** é uma etapa comum no pré-processamento de textos em tarefas de classificação e recuperação de informação. Stopwords são palavras muito frequentes em um idioma, como artigos, preposições e pronomes (ex.: “o”, “de”, “que”, “em”), que geralmente carregam pouco ou nenhum significado semântico relevante para a análise. A remoção dessas palavras visa reduzir o ruído nos dados e diminuir a dimensionalidade do espaço de representação textual, permitindo que os algoritmos de aprendizado de máquina foquem em termos mais informativos. No entanto, essa técnica deve ser aplicada com cuidado, pois em alguns contextos específicos, determinadas stopwords podem possuir valor semântico importante. Além disso, a lista de stopwords pode ser adaptada conforme o domínio ou objetivo do projeto, sendo possível incluir ou excluir termos conforme necessário (Manning, Raghavan & Schütze, 2008).

d) Vetorização

A **vetorização** é uma etapa essencial no processamento de linguagem natural, responsável por transformar textos em representações numéricas compreensíveis por algoritmos de aprendizado de máquina. Como os modelos computacionais não processam diretamente texto em linguagem natural, é necessário converter palavras ou documentos em vetores. Uma das abordagens mais simples e populares é o modelo de **bag-of-words (BoW)**, que representa o texto como uma matriz de frequência de termos, ignorando a ordem das palavras. Outra técnica amplamente utilizada é o **TF-IDF (Term Frequency-Inverse Document Frequency)**, que atribui pesos aos termos com base na frequência relativa em um documento e em todo o corpus, destacando os termos mais informativos. Mais recentemente, métodos de vetorização baseados em **word embeddings**, como Word2Vec, GloVe e BERT, passaram a ser adotados por sua capacidade de capturar relações semânticas e contextuais entre palavras em espaços vetoriais de alta dimensionalidade. A escolha da técnica de vetorização deve considerar fatores como a

complexidade do modelo, o volume de dados e a natureza da tarefa de classificação (Manning, Raghavan & Schütze, 2008).

2.1.2 Aplicação do modelo de machine learning

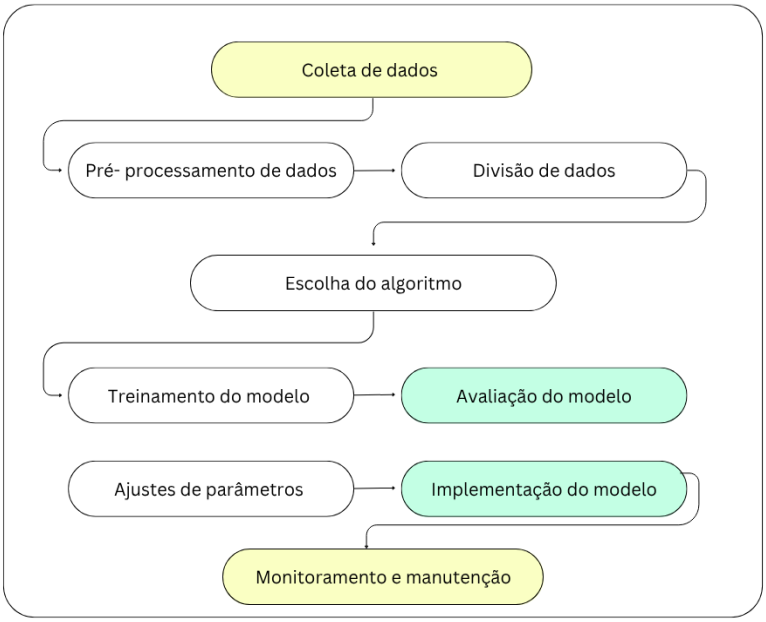
Após o pré-processamento do texto, que inclui etapas como tokenização, remoção de stopwords, stemming e vetorização, os dados já estruturados em formato numérico são utilizados para treinar um modelo de **aprendizado supervisionado**. Esse modelo tem como objetivo **aprender a associar padrões textuais às categorias predefinidas**, com base em exemplos rotulados previamente. Entre os algoritmos mais utilizados para essa tarefa estão o **Naive Bayes**, **Máquinas de Vetores de Suporte (SVM)**, **Árvores de Decisão** e modelos baseados em redes neurais, como os **classificadores com embeddings e deep learning**. Uma vez treinado, o modelo pode ser utilizado para **classificar automaticamente novos documentos**, atribuindo a eles uma ou mais categorias conforme seu conteúdo textual. A eficácia desse modelo é avaliada por métricas como **precisão, revocação, acurácia e F1-score**, que medem o quanto as classificações do modelo se aproximam da categorização ideal (Sebastiani, 2001; Manning, Raghavan & Schütze, 2008).

2.2 Treinamento supervisionado de modelos de machine learning

O *Machine Learning*, ou Aprendizado de Máquina, constitui-se como um dos principais ramos da Inteligência Artificial, cuja premissa fundamental reside na capacidade de sistemas computacionais extraírem conhecimento de conjuntos de dados, reconhecendo padrões e realizando predições de maneira autônoma. Trata-se, portanto, de uma abordagem em que algoritmos são habilitados a refinar seu desempenho progressivamente, a partir da experiência, sem que haja uma programação explícita para cada tarefa específica (Charleaux e Toledo, 2024). Para Müller (2017), o aprendizado de máquina confere aos sistemas a habilidade de aprender com a experiência, permitindo soluções dinâmicas e adaptativas em cenários diversos, como finanças, saúde, comércio e segurança.

O funcionamento do Machine Learning pode ser dividido em nove etapas. São elas apresentadas no fluxograma a seguir:

Figura 1- Funcionamento do Machine Learning



Fonte- Elaborado pelo autor.

Podendo ser dividido em quatro categorias principais, sendo: **Aprendizado supervisionado:** o algoritmo é treinado com dados rotulados, que já tem a resposta correta associada. O modelo aprende a fazer previsões e classificação sobre novos dados e pode ser aplicado em classificação de e-mails como spam; **Aprendizado não supervisionado:** o algoritmo aprende com dados não rotulados, buscando encontrar padrões e agrupamentos sem um conjunto de treinamento pré-definido. O modelo é usado, por exemplo, na segmentação de clientes em grupos com comportamentos semelhantes; **Aprendizado semi-supervisionado:** o algoritmo é treinado com dados rotulados e não rotulados, útil quando há uma quantidade limitada de dados rotulados e uma grande quantidade de dados não rotulados. O modelo pode ser aplicado como mais economia e eficiência na classificação de imagens médicas com poucos dados rotulados, por exemplo; **Aprendizado por reforço (Reinforcement Learning):** O algoritmo aprende por meio de interações com um ambiente, em um formato de tentativa e erro com sistema de recompensa. Ideal para treinamentos de um robô, por exemplo (Charleaux e Toledo, 2024).

Dentro do que foi apresentado, se destaca-se o aprendizado supervisionado, sendo considerado uma abordagem fundamental na área de Machine Learning, caracterizada pela presença de um conjunto de dados rotulado que serve como guia para o modelo durante o processo de treinamento. Essa metodologia é amplamente empregada em uma variedade de aplicações, desde reconhecimento de padrões até previsões e classificações (Ferreira, 2024).

O modelo é treinado para mapear as entradas para as saídas correspondentes com base nos exemplos fornecidos no conjunto de treinamento rotulado. Cada exemplo consiste em uma entrada associada a uma saída desejada, permitindo que o algoritmo ajuste seus parâmetros para fazer previsões precisas sobre novos dados não rotuldos (Ferreira, 2024).

Tipos de Tarefas:

Classificação: Uma das tarefas mais comuns no aprendizado supervisionado, onde o modelo aprende a atribuir rótulos a diferentes categorias. Exemplos incluem classificar e-mails como spam ou não spam, identificar objetos em imagens ou diagnosticar doenças com base em dados médicos.

Regressão: Utilizado quando a saída desejada é uma variável contínua. Por exemplo, prever o preço de uma casa com base em suas características, ou estimar a demanda por um produto em função de variáveis como preço e tempo.

Na prática pode ser aplicada na área da saúde: Diagnóstico médico com base em exames, previsão de doenças e personalização de tratamentos. Finanças: Avaliação de riscos, previsão de preços de ativos e detecção de fraudes. Marketing: Segmentação de clientes, personalização de campanhas e previsão de demanda.

Para Ferreira (2024), o aprendizado supervisionado permanece como uma estratégia central no campo da Inteligência Artificial, notadamente pelo seu impacto em diversos setores. Sua relevância advém da aptidão em interpretar e modelar estruturas complexas contidas em conjuntos de dados previamente rotulados, o que possibilita o desenvolvimento de soluções computacionais capazes de desempenhar tarefas avançadas com elevado grau de acurácia.

METODOLOGIA

Para a elaboração deste trabalho, foram utilizados métodos qualitativos com base em pesquisa bibliográfica. A investigação fundamentou-se na análise de artigos científicos, livros especializados e fontes acadêmicas confiáveis, permitindo a construção de um referencial teórico sólido e alinhado ao tema proposto. A seleção dos materiais foi feita considerando sua relevância, atualidade e contribuição para o aprofundamento do assunto abordado. O objetivo deste trabalho foi descrever a aplicação de machine learning para entender a intenção do usuário em uma solução de controle financeiro, que se baseou nas seguintes etapas:

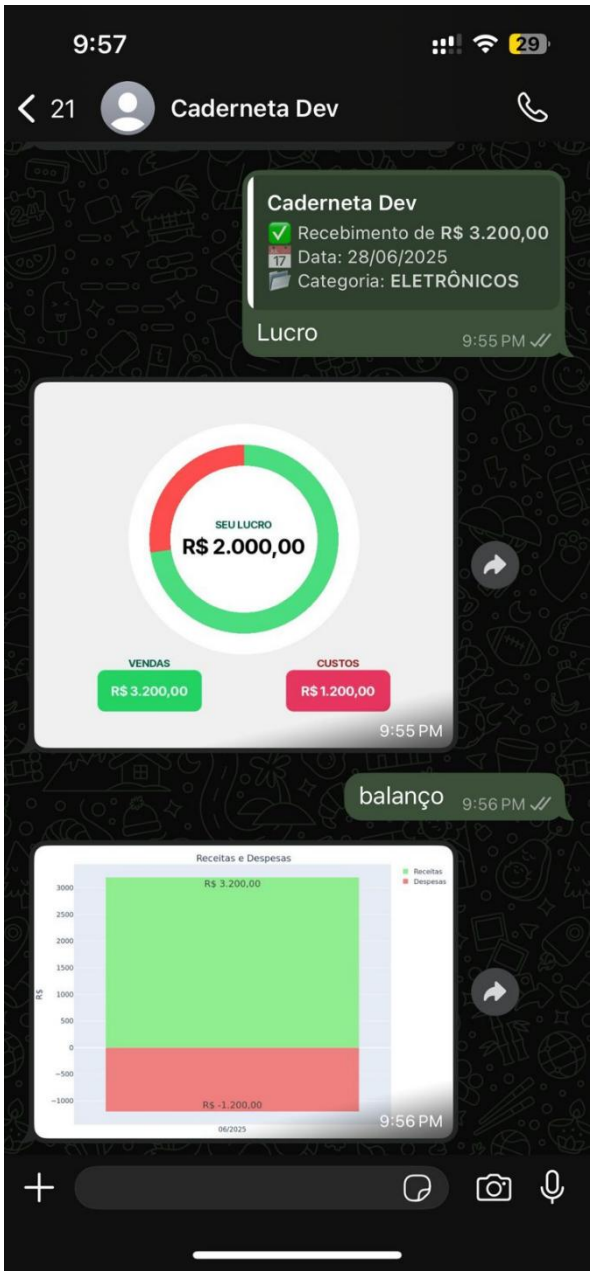
Coleta de dados dos usuários

A construção de uma solução de software começa com uma boa coleta de dados através de uma interface. O primeiro desafio nesse projeto foi a escolha de uma interface amigável, em que as interações com a solução ocorressem de forma fluida.

A escolha do WhatsApp para isso foi devido a dois motivos: a adoção do WhatsApp como aplicativo de mensagem padrão pelos brasileiros e a própria natureza do projeto: uma solução de controle financeiro por meio de linguagem natural. Ou seja, todo o projeto foi desenvolvido para que textos sejam interpretados e transformados em

comandos, sejam esses comandos, transações financeiras ou outros comandos do sistema como o comando “lucro” e “listar”.

Figura 2- Funcionamento controle financeiro via whatsapp



Fonte: Elaborado pelo autor

A classe “GerenciadorComandos” é responsável por registrar e processar os comandos do sistema. O registro dos comandos acontece por meio do padrão decorator do Python, que é uma função que modifica outra função sem alterá-la diretamente, como abaixo:

Figura 3- Funcionamento da Classe GerenciadorComandos

```python
class GerenciadorComandos:
    def __init__(self) -> None:
        self.commands: Dict[str, Comando] = {}
        self.prefix = "!"
        self.repo_transacao_leitura: RepoTransacaoLeitura = RepoTransacaoLeitura(session=get_session())

    def comando(
        self,
        name: str,
        description: str,
        icon: str = "",
        aliases: List[str] | None = None,
        oculto: bool = False,
    ) -> Callable[[Callable], Callable]:
        """Decorator to register commands"""

        def decorator(func: Callable):
            cmd = Comando(name, func, description, icon, aliases or [], oculto)
            self.registrar_comando(cmd)
            return func

        return decorator

    def registrar_comando(self, command: Comando):
        """Register a command and its aliases"""
        self.commands[command.name] = command
        for alias in command.aliases or []:
            self.commands[alias] = command
```

Figura 4- Segunda parte do funcionamento da Classe GerenciadorComandos

```python
from src.dominio.bot.entidade import GerenciadorComandos

bot = GerenciadorComandos()

@bot.comando("grafico fluxo", "Devolve gráfico de fluxo de caixa do mês", aliases=["grafico fluxo mm/aa"])
def grafico_fluxo(*args: List[str], **kwargs: Any) -> str:
    uploader = Uploader()
    usuario: Usuario = kwargs.get("usuario")
    intervalo = kwargs.get("intervalo") or intervalo_mes_atual()

    transacoes = bot.repo_transacao_leitura.buscar_por_intervalo_e_usuario_ordenando_por_data_e_valor(
        usuario_id=usuario.id, intervalo=intervalo
    )

    if not transacoes:
        return "Você ainda não registrou nenhuma despesa ou receita este mês"

    grafico = criar_grafico_fluxo_de_caixa(transacoes=transacoes)
    nome_arquivo = f"{grafico['nome_arquivo']}.png"
    caminho_arquivo: str = uploader.upload_file(nome_arquivo, grafico["dados"])
    return caminho_arquivo
```

Fonte: Elaborado pelo autor

Classificação de texto em transações

A principal funcionalidade da solução é a de registrar transações financeiras por meio de mensagem, como: “paguei 1200 aluguel” ou “recebi 2750 de salário 05/05”. Essa funcionalidade exige a análise de todo o texto para que a intenção do usuário seja entendida. Para isso, é aplicado um algoritmo de regressão logística que utiliza técnicas

de processamento de linguagem natural (NLP) para classificar automaticamente as mensagens como transações de débito ou crédito.

Figura 5- Funcionamento controle financeiro via whatsapp

Fonte: Elaborado pelo autor

O sistema implementa uma abordagem híbrida que combina aprendizado de máquina com regras baseadas em palavras-chave. O classificador utiliza um pipeline de processamento composto por um vetorizador TF-IDF (Term Frequency-Inverse Document Frequency) e um modelo de regressão logística treinado com dados

categorizados. O pré-processamento do texto inclui tokenização, remoção de stopwords em português e lematização para normalizar as palavras.

Figura 6- Parâmetros do classificador

```
class ClassificadorTexto:
    def __init__(self) -> None:
        self.csv_path = os.getenv("CSV_TREINAMENTO", "/opt/caderneta/static/dados_categorizados.csv")
        self.vectorizer_joblib = os.getenv("VECTORIZER_PATH", "/opt/caderneta/static/vectorizer.joblib")
        self.classifier_joblib = os.getenv("CLASSIFIER_PATH", "/opt/caderneta/static/classifier.joblib")
        self.vectorizer = (
            self._carregar_ou_criar_vetorizador(self.vectorizer_joblib)
            if self.vectorizer_joblib
            else TfidfVectorizer(max_features=100, ngram_range=(1, 2))
        )
        self.classifier = (
            self._carregar_ou_criar_classificador(self.classifier_joblib)
            if self.classifier_joblib
            else LogisticRegression(random_state=42)
        )
        self.pipeline = Pipeline([("vectorizer", self.vectorizer), ("classifier", self.classifier)])
        self.stop_words = set(stopwords.words("portuguese"))
        self.lemmatizer = WordNetLemmatizer()
        self.df = self._carregar_dataframe()
```

Fonte: Elaborado pelo autor

Para garantir robustez na classificação, o sistema mantém um conjunto de comandos predefinidos que incluem variações comuns de expressões financeiras. Para transações de débito, são reconhecidos termos como "paguei", "gastei", "compra", "pagamento", entre outros. Para transações de crédito, são identificados termos como "recebi", "vendi", "recebimento", "venda", etc. Quando a confiança do modelo de machine learning é inferior a 70%, o sistema recorre a essas regras baseadas em palavras-chave para determinar o tipo de transação.

Figura 7- Método classificador utilizando regressão logística

```python
def classificar_mensagem(self, mensagem: str) -> Tuple[str, Dict[str, float]]:
    try:
        mensagem_processada = self.pre_processar_texto(mensagem)

        previsao = self.pipeline.predict([mensagem_processada])[0]
        probabilidades = self.pipeline.predict_proba([mensagem_processada])[0]
        probs_dict = dict(zip(self.pipeline.classes_, probabilidades))

        return previsao, probs_dict

    except NotFittedError as erro:
        logger.info(f"Model not fitted yet: {erro}")
        self.treinar_modelo()
        self.salvar_modelo()
        return self.classificar_mensagem(mensagem)
```

Fonte: Elaborado pelo autor

Após a classificação, o sistema extrai automaticamente informações estruturadas da mensagem, incluindo valor monetário (com suporte a diferentes formatos brasileiros e internacionais), data, método de pagamento (PIX, cartão de crédito, débito, dinheiro, boleto, transferência), número de parcelas e categoria da transação. O parser é capaz de lidar com variações na estrutura das mensagens e normalizar valores monetários considerando diferentes separadores decimais e de milhares.

O modelo é treinado com dados reais de transações e pode ser atualizado incrementalmente, permitindo que o sistema melhore sua precisão ao longo do tempo. A implementação utiliza bibliotecas como scikit-learn para o aprendizado de máquina, NLTK para processamento de linguagem natural e pandas para manipulação de dados, garantindo eficiência e escalabilidade no processamento de mensagens financeiras.

Figura 8- Treinamento modelo com pipeline de TfidfVectorizer e LogisticRegression

```python
def treinar_modelo(self) -> str:
    """Train the model using the pipeline"""
    self.df["mensagem"] = self.df["mensagem"].apply(self.pre_processar_texto)

    X_train, X_test, y_train, y_test = train_test_split(
        self.df["mensagem"],
        self.df["classificacao"],
        test_size=0.3,
        random_state=42,
        shuffle=True,
    )

    self.pipeline.fit(X_train, y_train)

    y_pred = self.pipeline.predict(X_test)
    report: str = "\n" + classification_report(y_test, y_pred)
    return report
```

Fonte: Elaborado pelo autor

CONSIDERAÇÕES

Este trabalho apresentou uma solução de controle financeiro via WhatsApp, utilizando PLN e aprendizado de máquina para interpretar mensagens e registrar automaticamente transações. A escolha do WhatsApp como interface se mostrou adequada por sua ampla adoção e acessibilidade.

A combinação de vetorização TF-IDF com regressão logística e regras baseadas em palavras-chave resultou em uma classificação eficaz, mesmo com mensagens variadas. Além da aplicação técnica, o projeto destacou a importância de soluções simples e acessíveis para usuários com pouca familiaridade tecnológica.

A experiência proporcionou aprendizados valiosos sobre integração de técnicas de NLP e construção de interfaces intuitivas. Como continuidade, é possível explorar modelos mais avançados e ampliar as funcionalidades da plataforma.

REFERÊNCIAS BIBLIOGRÁFICAS

CHARLEAUX, L., TOLEDO, V. Machine Learning: o que é, como funciona e quais são os tipos de aprendizado de máquina Tecnoblog. Disponível em: <https://tecnoblog.net/responde/machine-learning-o-que-e-como-funciona-e-quais-sao-os-tipos-de-aprendizado-de-maquina/>. Acessado em: 22 de junho de 2025.

BATISTA, R., ARAÚJO, S. Introdução à programação orientada a objetos*. Teresina: Instituto Federal de Educação, Ciência e Tecnologia do Piauí, 2013. 118 p.

DASARI, V.; RAO, G. S. V. P. R. Text categorization techniques: A literature survey. International Journal of Computer Science and Engineering, v. 1, n. 2, p. 87–100, 2012.

GRONER,. Estrutura de dados e algoritmos com JavaScript. 1. ed. Rio de Janeiro: Novatec, 2019. 408 p.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. Introduction to Information Retrieval. Cambridge: Cambridge University Press, 2008.

MARTIN, R, C. Arquitetura limpa: o guia do artesão para estrutura e design de software*. Tradução de Guilherme G. Milani. 1. ed. Rio de Janeiro: Alta Books, 2018. 431 p.

MARTIN, R, C. Código limpo: habilidades práticas do Agile Software*. Tradução de Guilherme G. Milani. 1. ed. Rio de Janeiro: Alta Books, 2009. 425 p.

MÜLLER, A, C.; GUIDO, S. Introdução ao machine learning com Python: um guia para cientistas de dados. Tradução de Rafael Gomes. 1. ed. São Paulo: Novatec, 2017. 376 p.

SEBASTIANI, F. Machine learning in automated text categorization. ACM Computing Surveys, v. 34, n. 1, p. 1–47, 2001. DOI: 10.1145/505282.505283