

Vitor Gomes Reis

**AGENTE INTELIGENTE PARA
QUALIFICAÇÃO DE *LEADS* NO SETOR
AUTOMOTIVO UTILIZANDO *MACHINE
LEARNING***

Uberlândia – MG

2025

Vitor Gomes Reis

**AGENTE INTELIGENTE PARA QUALIFICAÇÃO DE
LEADS NO SETOR AUTOMOTIVO UTILIZANDO
*MACHINE LEARNING***

Universidade Federal de Uberlândia – UFU

Faculdade de Computação – FACOM

Graduação em Ciência da Computação

Orientador: Marcia Aparecida Fernandes

Uberlândia – MG

2025

RESUMO

Este Trabalho de Conclusão de Curso detalha o desenvolvimento, implementação e comparação de modelos de *Machine Learning* para a qualificação inteligente de leads, potenciais clientes que demonstraram interesse inicial através de canais digitais, em uma concessionária de veículos de alto padrão, através de um agente de Inteligência Artificial denominado “Betina”. Motivada pela necessidade de otimizar o atendimento frente a uma crescente demanda digital e a sobrecarga da equipe de vendas, a solução evoluiu de um *Minimum Viable Product* em plataforma *no-code* (Zaia) para uma arquitetura robusta integrando Make.com, ChatGPT-4o e uma API de ML customizada. O núcleo do trabalho reside na comparação de duas abordagens preditivas treinadas com 72 conversas históricas: (1) um classificador Random Forest utilizando features TF-IDF e de comportamento, interpretado via Modelo Surrogado; e (2) o fine-tuning de modelos Transformer (BERT e mBERT). A metodologia abrangeu pré-processamento avançado, incluindo separação de falas e anonimização. Enquanto o fine-tuning de BERT demonstrou potencial de alta acurácia, limitações computacionais impediram sua conclusão. O Random Forest, embora com desempenho moderado, provou-se viável e interpretável, fornecendo insights acionáveis. O projeto valida sua eficácia através de uma análise empírica de casos reais e da extração de insights estratégicos para a equipe de vendas com os leads qualificados pela abordagem viável, demonstrando a aplicação prática e os desafios de diferentes técnicas de IA e ML na otimização de processos comerciais.

ABSTRACT

This Final Year Project details the development, implementation, and comparison of Machine Learning models for intelligent lead - potential customers who have demonstrated initial interest through digital channels - qualification at a luxury car dealership, facilitated by an Artificial Intelligence agent named “Betina”. Motivated by the need to optimize customer service amidst growing digital demand and the consequent overload of the sales team, the solution evolved from a Minimum Viable Product (MVP) on a no-code platform (Zaia) to a robust architecture integrating Make.com, ChatGPT-4o, and a custom ML API. The core of this study lies in the comparison of two predictive approaches trained on a dataset of 72 historical conversations: (1) a Random Forest classifier utilizing TF-IDF and behavioral features, interpreted via a surrogate model; and (2) the fine-tuning of Transformer models (BERT and mBERT). The methodology encompassed advanced preprocessing, including speaker separation and data anonymization. While the fine-tuning of BERT demonstrated potential for high accuracy, computational constraints limited its viability in this context. Conversely, the Random Forest model proved to be a robust and interpretable solution, yielding actionable insights. The project validates the system’s effectiveness through empirical case analysis and the generation of strategic insights for the sales team regarding qualified leads, demonstrating the practical application and challenges of deploying AI and ML techniques to optimize business processes.

LISTA DE ILUSTRAÇÕES

Figura 1 – Diagrama Betina Completo	29
Figura 2 – Diagrama do Fluxo Principal de Recebimento de Mensagens (Recebe Betina) no Make.com	31
Figura 3 – Diagrama do Fluxo de Continuidade e Envio de Mensagem (Envia Betina) no Make.com	31
Figura 4 – Diagrama do fluxo "Acessar Estoque" para consulta em XML no Make.com	33
Figura 5 – Diagrama do fluxo "Atualizar Estoque" via XML no Make.com	33
Figura 6 – Conversa mais promissora da Betina	48
Figura 7 – Segunda conversa mais promissora da Betina	49

LISTA DE TABELAS

Tabela 1 – Relatório de Classificação do Modelo Random Forest (Validação Cruzada k=3)	41
Tabela 2 – Matriz de Confusão Agregada do Modelo Random Forest (Validação Cruzada k=3)	42
Tabela 3 – Relatório de Classificação do Modelo BERT (Conjunto de Teste) . . .	42
Tabela 4 – Relatório de Classificação do Modelo mBERT (Conjunto de Teste) . .	43
Tabela 5 – Comparação de desempenho: Random Forest vs. BERT Base	43
Tabela 6 – Impacto das Features de Comportamento na Venda	44
Tabela 7 – Principais Termos Linguísticos de Sucesso (Venda)	45
Tabela 8 – Principais Termos Linguísticos de Insucesso (Não Venda)	45

SUMÁRIO

	Sumário	6
1	INTRODUÇÃO	9
1.1	OBJETIVOS	10
1.1.1	OBJETIVO GERAL	10
1.1.2	OBJETIVOS ESPECÍFICOS	10
1.2	JUSTIFICATIVA	11
	Declaração de Uso de Inteligência Artificial	11
2	FUNDAMENTAÇÃO TEÓRICA	12
2.1	INTELIGÊNCIA ARTIFICIAL (IA)	12
2.1.1	PROCESSAMENTO DE LINGUAGEM NATURAL (PLN)	12
2.1.2	CHATBOTS E FERRAMENTAS DE CONSTRUÇÃO DE AGENTES	13
2.2	APRENDIZADO DE MÁQUINA (MACHINE LEARNING)	13
2.2.1	VETORIZAÇÃO DE TEXTO: TF-IDF	13
2.2.2	RANDOM FOREST (FLORESTA ALEATÓRIA)	14
2.2.3	VALIDAÇÃO DE MODELOS: VALIDAÇÃO CRUZADA	14
2.2.4	MODELOS TRANSFORMER (BERT/MBERT)	15
2.2.5	FINE-TUNING (AJUSTE FINO)	15
2.2.6	EXPLICABILIDADE DE MODELOS: MODELOS SURROGADOS	15
3	TRABALHOS RELACIONADOS	17
3.1	O SETOR AUTOMOTIVO E A DIGITALIZAÇÃO: ANÁLISE DE MONTEIRO (2023)	17
3.2	CHATBOTS E AUTOMAÇÃO DO ATENDIMENTO: ANÁLISE DE PACHECO (2021)	18
3.3	PREDIÇÃO DE SATISFAÇÃO A PARTIR DE LOGS DE CONVERSA: ANÁLISE DE AY ET AL. (2025)	19
4	DESENVOLVIMENTO DA SOLUÇÃO	21
4.1	FASE 1: PRODUTO MÍNIMO VIÁVEL (MVP) NA PLATAFORMA ZAIA	21
4.1.1	ANÁLISE COMPARATIVA E SELEÇÃO DA PLATAFORMA	21
4.1.2	ARQUITETURA E IMPLEMENTAÇÃO INICIAL	22
4.1.3	DESENVOLVIMENTO ITERATIVO DO AGENTE CONVERSACIONAL	22
4.1.3.1	O Desafio da Qualificação Eficiente	22

4.1.3.2	Definição do Fluxo de Conversa Estruturado	22
4.1.3.3	Engenharia de Prompt e Refinamento de Regras	23
4.1.3.4	Prompt Final Utilizado na Zaia e no Make	24
4.2	FASE 2: ARQUITETURA FINAL COM MAKE.COM E MACHINE LEARNING	28
4.2.1	Justificativa e Seleção da Plataforma Make.com	28
4.2.2	Arquitetura da Solução Final	29
4.2.3	DETALHAMENTO DO FLUXO OPERACIONAL NO MAKE.COM	30
4.2.3.1	Fluxo de Recepção e Resposta (Fluxo Recebe/Envia Betina)	31
4.2.3.2	Fluxos de Gerenciamento de Dados (Estoque)	33
4.2.4	IMPLEMENTAÇÃO DA LÓGICA DE QUALIFICAÇÃO PREDITIVA	33
4.3	MODELO DE MACHINE LEARNING	34
4.3.1	COLETA E PREPARAÇÃO DO DATASET	34
4.3.1.1	Fonte de Dados e Amostragem	34
4.3.1.2	Tratamento de Desbalanceamento de Classes	35
4.3.2	ENGENHARIA E PRÉ-PROCESSAMENTO DE FEATURES	35
4.3.2.1	Extração e Separação de Falas	35
4.3.2.2	Criação de Features de Comportamento	35
4.3.2.3	Pipeline de Pré-processamento de Texto	36
4.3.3	MODELAGEM PREDITIVA	37
4.3.3.1	Abordagem 1: Random Forest e TF-IDF	37
4.3.3.1.1	Vetorização e Construção da Matriz de Features	37
4.3.3.1.2	Seleção e Evolução do Algoritmo	37
4.3.3.1.3	Validação do Modelo	37
4.3.3.1.4	Técnica de Interpretação: O Modelo Surrogado	38
4.3.3.2	Abordagem B: Fine-tuning de Modelos Transformer (BERT/mBERT)	38
5	RESULTADOS E DISCUSSÃO	41
5.1	desempenho PREDITIVO COMPARATIVO	41
5.1.1	ABORDAGEM A: desempenho DO RANDOM FOREST (MODELO VIÁVEL)	41
5.1.2	ABORDAGEM B: desempenho DOS MODELOS TRANSFORMER (BERT/M-BERT)	42
5.1.2.1	desempenho do BERT (bert-base-uncased)	42
5.1.2.2	desempenho do mBERT (bert-base-multilingual-cased)	43
5.2	DISCUSSÃO DO desempenho E SELEÇÃO DO MODELO FINAL	43
5.3	ANÁLISE DE INSIGHTS DO MODELO PREDITIVO (RANDOM FOREST + SURROGADO)	44
5.3.1	ANÁLISE DOS PADRÕES COMPORTAMENTAIS	44
5.3.2	ANÁLISE DOS PADRÕES LINGÜÍSTICOS	45
5.4	ANÁLISE QUALITATIVA DE CASOS (ESTUDO DE CONVERSAS)	46

5.4.1	ESTUDO DE CASO 1: A CONVERSA MAIS PROMISSORA	46
5.4.2	ESTUDO DE CASO 2: A CONVERSA MENOS PROMISSORA	46
5.5	VALIDAÇÃO EMPÍRICA E COMPORTAMENTO EM CAMPO . . .	46
5.5.1	RESULTADOS DA PREDIÇÃO EM NOVO DATASET	46
5.5.2	DISCUSSÃO SOBRE O COMPORTAMENTO DE FÁBRICA	47
5.5.2.1	Análise Qualitativa das Conversas Mais Promissoras em Campo	47
5.5.3	CASO EMPÍRICO 1: MAIOR PROBABILIDADE (36%)	48
5.5.4	CASO EMPÍRICO 2: SEGUNDA MAIOR PROBABILIDADE (24%)	49
5.6	VALIDAÇÃO DA SOLUÇÃO NO CONTEXTO DE NEGÓCIO . . .	49
6	CONCLUSÃO E TRABALHOS FUTUROS	51
6.1	LIMITAÇÕES DO ESTUDO	51
6.2	TRABALHOS FUTUROS	52
	REFERÊNCIAS	54
	APÊNDICE A – CÓDIGO FONTE DO MODELO PREDITIVO (ABORDAGEM RANDOM FOREST)	56
	APÊNDICE B – CÓDIGO FONTE DA ABORDAGEM COMPARATIVA (BERT/MBERT)	65

1 INTRODUÇÃO

O mercado de veículos de alto padrão no Brasil apresenta um cenário altamente competitivo e dinâmico, onde a excelência na experiência do cliente e a eficiência operacional no atendimento são fatores cruciais e distintivos para o sucesso das empresas. A concessionária Pax Concept, reconhecendo a necessidade de inovar suas estratégias de captação e gestão de leads (um lead é uma pessoa que demonstrou interesse em um produto e forneceu seus dados de contato), investiu significativamente em um novo canal de aquisição: o tráfego pago, por meio de plataformas consolidadas como o Meta Ads. Esta iniciativa, embora extremamente bem-sucedida em gerar um volume crescente e qualificado de contatos, prontamente expôs um gargalo operacional crítico no fluxo de trabalho: a limitação intrínseca da equipe de vendas em gerenciar simultaneamente o atendimento presencial na loja e a demanda exponencial advinda dos canais digitais.

A equipe de vendas, composta pelos mesmos colaboradores que atuam em ambas as frentes de atendimento, enfrentava um conflito de atenção direto, o que frequentemente resultava em um tempo de resposta consideravelmente lento para os leads digitais — um fator determinante e prejudicial na conversão de vendas online. A alternativa de simplesmente expandir a equipe de vendas foi cuidadosamente analisada, mas preterida em favor de uma solução tecnológica inovadora que oferecesse não apenas a resolução do problema pontual de sobrecarga, mas também escalabilidade inerente e inteligência estratégica ao processo de pré-vendas. Neste contexto estratégico, foi concebido o agente de Inteligência Artificial “Betina”, um *chatbot* meticulosamente projetado para otimizar integralmente o processo de qualificação e atendimento inicial de leads.

Inicialmente implementado na plataforma *no-code* Zaia e integrada diretamente ao WhatsApp, o agente “Betina” assumiu a linha de frente para os leads gerados pelos anúncios, oferecendo atendimento instantâneo e contínuo (24/7) e realizando uma qualificação preliminar. Os resultados obtidos nesta fase foram notavelmente satisfatórios, permitindo a otimização da equipe humana de vendas, com uma redução de quatro para três vendedores, concomitantemente a um aumento percebido na produtividade. Contudo, como é comum em soluções *no-code* escaláveis, a plataforma inicial apresentou limitações intrínsecas de customização e flexibilidade, além de um custo operacional que motivou a evolução do projeto para uma arquitetura mais robusta, proprietária e, a longo prazo, economicamente mais vantajosa, utilizando a plataforma de automação Make.com como orquestrador central.

A nova fase de desenvolvimento do projeto não se limitou a replicar a funcionalidade anterior de atendimento. Para aprimorar significativamente a capacidade de qualificação

do agente e transformá-lo em uma ferramenta preditiva, foi desenvolvido um modelo de *Machine Learning* avançado, treinado com um volume significativo de dados históricos de conversas reais da própria concessionária. Este modelo preditivo analisa padrões linguísticos e comportamentais manifestados nas interações para metrificar, com precisão, a probabilidade de uma conversa específica resultar em uma venda. Tal capacidade permite que o agente “Betina” transfira para um vendedor humano apenas os leads mais promissores e já qualificados, no momento mais oportuno e estratégico, maximizando a eficiência da equipe de vendas.

1.1 OBJETIVOS

A concepção deste trabalho é guiada por um objetivo geral claro, que se desdobra em objetivos específicos para garantir a execução e a validação da solução proposta.

1.1.1 OBJETIVO GERAL

O objetivo geral deste trabalho é detalhar o desenvolvimento e a implementação de um agente de Inteligência Artificial avançado para o processo de pré-vendas, que utiliza um modelo preditivo de *Machine Learning* para qualificar leads de maneira eficiente, otimizar o fluxo de atendimento ao cliente e, por consequência, aumentar o potencial de conversão de vendas da concessionária de luxo.

1.1.2 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral, este trabalho se propõe a cumprir os seguintes objetivos específicos:

- Analisar e documentar a evolução completa da arquitetura do agente inteligente, desde sua concepção inicial na plataforma Zaia até a solução final integrada com Make.com e uma API customizada.
- Detalhar minuciosamente a metodologia de coleta, pré-processamento, engenharia de prompt.
- Implementar e validar duas abordagens distintas de *Machine Learning*: (1) um modelo Random Forest com features TF-IDF e de comportamento; (2) o fine-tuning de modelos Transformer (BERT/mBERT).
- Comparar o desempenho preditivo e a viabilidade de implementação das duas abordagens no contexto do projeto.
- Avaliar o comportamento do modelo em ambiente de produção e extrair *insights* interpretáveis que auxiliem a estratégia da equipe de vendas.

1.2 JUSTIFICATIVA

A relevância deste projeto se fundamenta em três pilares principais: a justificativa operacional, que aborda a resolução de um problema de negócio imediato; a justificativa tecnológica e econômica, que posiciona a solução frente a alternativas de mercado; e a justificativa estratégica, que contempla o potencial de escalabilidade do produto final.

Operacionalmente, a principal justificativa reside na otimização do processo de atendimento. Conforme validado por Lima (LIMA, 2022), a redução do tempo de resposta é um fator crítico para o sucesso de vendas no setor automotivo. O agente Betina ataca diretamente este ponto, oferecendo disponibilidade 24/7 e garantindo que nenhum lead gerado por tráfego pago fique sem um primeiro contato imediato, independentemente da disponibilidade da equipe de vendas presencial.

Do ponto de vista tecnológico e econômico, o desenvolvimento de uma solução proprietária se mostrou a decisão mais acertada. No início do projeto, uma prospecção de mercado não revelou soluções consolidadas que aplicassem um modelo preditivo de *Machine Learning* para a qualificação de leads especificamente no nicho de veículos de luxo via WhatsApp no Brasil, o que conferiu um caráter inovador à iniciativa. Posteriormente, identificou-se a existência de uma plataforma emergente, a IAuto, que oferece funcionalidades análogas. Contudo, uma análise de viabilidade revelou que os custos de licenciamento da solução externa eram substancialmente superiores aos custos projetados para o desenvolvimento interno. Isso validou a decisão de desenvolver a ferramenta, não apenas como um exercício acadêmico, mas como uma iniciativa com claro retorno sobre o investimento (ROI), alinhando a inovação tecnológica à sustentabilidade financeira.

Finalmente, a justificativa estratégica do projeto reside no seu potencial de escalabilidade. A concepção da arquitetura, que desacopla o orquestrador de fluxo (Make.com) do motor de inteligência (a API de Machine Learning), foi projetada de forma modular. Esta abordagem de microsserviços permite que, no futuro, a solução de análise preditiva seja ofertada como um serviço independente (*Software as a Service* - SaaS) para outras concessionárias do segmento, transformando um projeto de otimização interna em um potencial produto de mercado.

DECLARAÇÃO DE USO DE INTELIGÊNCIA ARTIFICIAL

Durante a elaboração deste trabalho, foram utilizadas ferramentas de Inteligência Artificial Generativa como apoio à produtividade. O modelo Google Gemini foi empregado para revisão gramatical e suporte na codificação (L^AT_EX e Python). A autoria intelectual, bem como a responsabilidade pela veracidade dos dados e das conclusões aqui expostas, permanece integralmente com o autor.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos fundamentais que servem como base teórica para o desenvolvimento e a compreensão do agente inteligente proposto. A estrutura desta fundamentação visa prover um arcabouço sólido, detalhando desde os princípios gerais da Inteligência Artificial e Processamento de Linguagem Natural até as especificidades das arquiteturas de sistemas, metodologias de desenvolvimento e as técnicas avançadas de Aprendizado de Máquina empregadas no núcleo deste projeto.

2.1 INTELIGÊNCIA ARTIFICIAL (IA)

Inteligência Artificial (IA) é um campo multidisciplinar da ciência da computação dedicado ao desenvolvimento de sistemas capazes de simular e, em alguns aspectos, superar capacidades cognitivas humanas. Estes sistemas são projetados para aprender a partir de dados, raciocinar logicamente, resolver problemas complexos e tomar decisões autônomas (RUSSELL; NORVIG, 2020). A IA abrange um vasto espectro de subáreas, incluindo, mas não se limitando a, aprendizado de máquina, redes neurais, visão computacional e processamento de linguagem natural (GOODFELLOW; BENGIO; COURVILLE, 2016). Embora o conceito de IA tenha suas raízes no século XX, os avanços mais expressivos, particularmente nas últimas décadas, foram impulsionados pelo desenvolvimento do aprendizado profundo (*deep learning*) e pela emergência da IA generativa. No contexto do presente trabalho, a IA é o pilar que habilita o agente “Betina” a interagir de forma inteligente e a qualificar leads.

2.1.1 PROCESSAMENTO DE LINGUAGEM NATURAL (PLN)

O Processamento de Linguagem Natural (PLN) representa uma subárea da Inteligência Artificial que se dedica à interação entre computadores e a linguagem humana. O objetivo primordial do PLN é capacitar sistemas computacionais a compreender, interpretar e gerar texto de forma semanticamente significativa e contextualmente relevante (LIDDY, 2001). As aplicações do PLN são vastas e incluem tarefas como análise de sentimentos, tradução automática, e reconhecimento de entidades nomeadas (BIRD; KLEIN; LOPER, 2009). A evolução histórica do PLN progrediu de abordagens baseadas em regras heurísticas para modelos estatísticos mais sofisticados e, mais recentemente, culminou no advento de arquiteturas de redes neurais profundas, como BERT e GPT, que estabeleceram novos paradigmas de desempenho e capacidades no tratamento da linguagem humana. No escopo deste trabalho, o PLN é fundamental para que o agente “Betina” possa analisar

e compreender as nuances das conversas textuais com os clientes, extraindo informações cruciais para a qualificação de leads.

2.1.2 CHATBOTS E FERRAMENTAS DE CONSTRUÇÃO DE AGENTES

Um *chatbot* é um programa de computador projetado para simular uma conversação com usuários humanos através de interfaces textuais ou de voz (SHAWAR; ATWELL, 2007). Essas ferramentas visam automatizar o atendimento e a interação, liberando recursos humanos para tarefas mais complexas. As plataformas e ferramentas para a construção de agentes conversacionais facilitam a criação e gestão desses sistemas, oferecendo recursos essenciais como gerenciamento de diálogos, integrações com outras plataformas e funcionalidades analíticas (GAO; GALLEY; LI, 2019). A evolução dessas ferramentas, de sistemas complexos baseados em código para plataformas *low-code/no-code* como Zaia, Make, Langflow e N8N, democratizou o desenvolvimento de agentes conversacionais (BURNS; NAULT, 2021).

2.2 APRENDIZADO DE MÁQUINA (MACHINE LEARNING)

Aprendizado de Máquina (*Machine Learning* - ML) é uma subárea da Inteligência Artificial que se concentra no desenvolvimento de algoritmos que capacitam os computadores a aprender a partir de dados, sem a necessidade de serem explicitamente programados para cada tarefa (RUSSELL; NORVIG, 2020). Em essência, os modelos de ML identificam padrões e relações nos dados, utilizando-os para realizar previsões ou tomar decisões. No contexto deste trabalho, o **Aprendizado Supervisionado** foi a abordagem central. Nesta modalidade, o modelo é treinado com um conjunto de dados rotulados, onde cada amostra de entrada possui uma saída correspondente conhecida (neste caso, conversas classificadas como “venda” ou “não venda”), permitindo que o modelo aprenda a mapear entradas para saídas e, subsequentemente, faça previsões sobre novos dados não vistos.

2.2.1 VETORIZAÇÃO DE TEXTO: TF-IDF

Algoritmos de Aprendizado de Máquina, por sua natureza matemática, não processam texto diretamente; eles requerem uma representação numérica. A técnica de **TF-IDF** (*Term Frequency-Inverse Document Frequency*) foi empregada neste projeto para realizar essa conversão fundamental (SALTON; YANG; YU, 1975). O processo de vetorização com TF-IDF pode ser compreendido em duas partes:

- **TF (Frequência do Termo):** Mede a importância de uma palavra dentro de um único documento (uma conversa). É calculada pela frequência com que um termo

aparece na conversa. A intuição é que palavras repetidas com frequência em um diálogo são importantes para o contexto daquele diálogo específico.

- **IDF (Frequência Inversa do Documento):** Mede a importância de uma palavra em todo o conjunto de dados (todas as conversas). Palavras que são muito comuns em todos os documentos (e.g., “carro”, “olá”) são consideradas menos informativas e recebem um peso baixo. Palavras que são raras e aparecem em poucas conversas (e.g., “contrato”, “financiamento”) são consideradas mais distintivas e recebem um peso alto.

O score final TF-IDF para uma palavra em uma conversa é o produto de seu TF e seu IDF. **A escolha do TF-IDF para este projeto se justifica por sua capacidade de automaticamente filtrar o ruído de palavras comuns e destacar os termos que são verdadeiramente distintivos e potencialmente preditivos para uma negociação específica**, fornecendo um sinal claro e ponderado para o modelo de aprendizado.

2.2.2 RANDOM FOREST (FLORESTA ALEATÓRIA)

O **Random Forest** é um algoritmo de *ensemble learning* que se destaca pela sua robustez e versatilidade em tarefas de classificação e regressão (BREIMAN, 2001). Ele opera construindo uma vasta quantidade de árvores de decisão individuais de forma independente durante a fase de treinamento. Para uma tarefa de classificação, como a qualificação de leads, o resultado final é determinado pela classe que obtém a maioria dos votos entre as previsões de todas as árvores.

A principal justificativa para a escolha do Random Forest neste projeto, em detrimento de modelos lineares como a Regressão Logística, reside em sua notável capacidade de modelar relações **não-lineares** e de capturar **interações complexas** entre as *features*. A linguagem natural é inerentemente contextual; o significado de uma palavra pode ser alterado pela presença de outra. O Random Forest pode aprender regras como: “*SE a palavra 'não' está presente E a palavra 'gostei' também, ENTÃO o resultado tende a ser negativo*”, uma nuance que modelos lineares têm dificuldade em capturar. Essa capacidade de entender o contexto torna o Random Forest uma ferramenta superior para analisar a complexidade de um diálogo de vendas.

2.2.3 VALIDAÇÃO DE MODELOS: VALIDAÇÃO CRUZADA

Para obter uma estimativa robusta, confiável e imparcial do desempenho generalizável do modelo preditivo, foi empregada a metodologia de **Validação Cruzada Estratificada** (*Stratified K-Fold Cross-Validation*) (KOHAVI, 1995). Uma simples

divisão entre dados de treino e teste pode levar a resultados enviesados pela sorte na amostragem. A validação cruzada mitiga esse risco ao dividir o conjunto de dados em 'k' subconjuntos (neste projeto, $k=3$). O modelo é então treinado e avaliado 'k' vezes, onde a cada iteração, um subconjunto diferente é usado para teste e os demais para treino.

Adicionalmente, a abordagem Estratificada foi crucial. Dado o desbalanceamento de classes no nosso dataset (mais “não-vendas” do que “vendas”), a estratificação garante que a proporção original das classes seja mantida em cada um dos 'k' subconjuntos. **Esta metodologia foi escolhida por seu rigor acadêmico, garantindo que as métricas de desempenho reportadas (como a acurácia média) sejam uma representação fidedigna da capacidade do modelo de operar em dados do mundo real.**

2.2.4 MODELOS TRANSFORMER (BERT/MBERT)

Os modelos **Transformer**, como **BERT** (*Bidirectional Encoder Representations from Transformers*) (DEVLIN et al., 2019) e sua versão multilíngue **mBERT**, representam o estado da arte em PLN. Sua arquitetura inovadora, baseada em mecanismos de **auto-atenção**, permite que o modelo processe a sequência de texto inteira simultaneamente, considerando o contexto de cada palavra de forma bidirecional (olhando para palavras anteriores e posteriores). Isso resulta em uma compreensão contextual e semântica muito mais profunda do que abordagens como TF-IDF. Eles geram representações numéricas (embeddings) para cada palavra que são dinâmicas e dependentes do contexto da frase.

2.2.5 FINE-TUNING (AJUSTE FINO)

O **Fine-tuning** é uma técnica de *Transfer Learning* aplicada a modelos pré-treinados como BERT/mBERT. Consiste em pegar o modelo, que já possui um vasto conhecimento linguístico adquirido em bilhões de textos, e continuar seu treinamento com um dataset menor e específico da tarefa desejada (nossas 72 conversas). Uma camada de classificação é adicionada ao topo do modelo pré-treinado, e os pesos de parte ou de todo o modelo são ajustados levemente com os dados específicos. Isso permite especializar o modelo para a tarefa (classificar vendas) sem perder seu conhecimento geral, sendo uma forma eficaz de obter alto desempenho mesmo com dados limitados.

2.2.6 EXPLICABILIDADE DE MODELOS: MODELOS SURROGADOS

Modelos de Aprendizado de Máquina complexos, como o Random Forest, são frequentemente referidos como “caixas-pretas” devido à dificuldade intrínseca de interpretar diretamente como suas decisões são tomadas. No contexto de um TCC onde a extração de insights acionáveis é um objetivo central, a interpretabilidade é fundamental. Para contornar este desafio, e após encontrar limitações de memória e compatibilidade com outras técnicas

como o SHAP, foi implementada a técnica de **Modelo Surrogado** (RIBEIRO; SINGH; GUESTRIN, 2016).

Nesta abordagem, um modelo mais simples e inerentemente interpretável (neste caso, uma Regressão Logística) foi treinado com um objetivo diferente: em vez de prever o resultado real da venda, ele foi treinado para imitar as previsões de probabilidade do modelo Random Forest principal. Os coeficientes obtidos do modelo surrogado, por serem diretamente interpretáveis (positivos ou negativos), foram então analisados para inferir os padrões que o modelo complexo aprendeu. **Esta técnica representa uma solução elegante para o dilema entre desempenho e interpretabilidade, permitindo-nos utilizar o poder preditivo do Random Forest e, ao mesmo tempo, extrair os insights claros e direcionais necessários para entender a lógica por trás da qualificação de leads.**

3 TRABALHOS RELACIONADOS

Este capítulo apresenta uma revisão de trabalhos acadêmicos relevantes que estabelecem o contexto e fornecem a base de conhecimento para o desenvolvimento do agente inteligente “Betina”. Cada estudo é criteriosamente contextualizado e conectado diretamente aos objetivos, às metodologias empregadas e às inovações propostas neste trabalho. O intuito é demonstrar como a presente pesquisa se insere no panorama atual da Ciência da Computação e da Gestão, e como contribui para o campo do atendimento digital e da qualificação de leads no setor automotivo de luxo.

3.1 O SETOR AUTOMOTIVO E A DIGITALIZAÇÃO: ANÁLISE DE MONTEIRO (2023)

Monteiro ([MONTEIRO, 2023](#)), em sua dissertação intitulada “Inovação e Digitalização: Vantagens da implementação das lojas online no setor automóvel”, investiga profundamente a transformação digital que permeia o setor automotivo contemporâneo. O estudo revela que, embora a implementação de lojas online ofereça vantagens inegáveis como agilidade nas transações e ampliação do alcance de mercado, fatores como a confiança do consumidor e a percepção de risco na compra de veículos online representam obstáculos significativos à sua plena adoção. A autora conclui que a digitalização pode, de fato, gerar um impacto positivo, contanto que as empresas consigam desenvolver e implementar estratégias eficazes para transmitir confiança e segurança aos consumidores durante todo o processo de aquisição.

O trabalho de Monteiro ([MONTEIRO, 2023](#)) estabelece um arcabouço fundamental e o cenário macro no qual o projeto Betina se insere e adquire sua relevância: a crescente e irreversível digitalização do processo de compra de veículos, inclusive nos segmentos de luxo. A ênfase da autora na imperatividade de construir confiança e na necessidade de mitigar a percepção de risco no ambiente digital ressoa diretamente com a proposição de valor do agente “Betina”. Ao atuar como uma linha de frente eficiente e inteligente, proporcionando um primeiro contato qualificado e “humanizado” através da inteligência artificial, o projeto “Betina” visa não apenas otimizar o fluxo operacional de qualificação de leads, mas também edificar a credibilidade inicial necessária para que o cliente se sinta seguro e confiante em prosseguir para uma interação mais aprofundada com um vendedor humano e, em última instância, a concretização da compra. Desta forma, o presente trabalho valida a premissa de que a digitalização deve ser concomitantemente acompanhada por mecanismos que solidifiquem a confiança do consumidor.

Lima ([LIMA, 2022](#)), em sua dissertação “Os desafios da omnicanalidade no setor automotivo: a influência do atendimento em plataformas digitais sobre o desempenho de vendas de veículos”, aborda a complexa realidade da omnicanalidade e investiga profundamente a influência crítica do atendimento em plataformas digitais sobre o desempenho de vendas de veículos. Utilizando uma abordagem de pesquisa eminentemente quantitativa, o autor demonstra de forma inequívoca que a redução do tempo médio de resposta nos canais digitais exerce um efeito causal e significativamente positivo no Net Promoter Score (NPS). Por sua vez, o aumento do NPS correlaciona-se positivamente com a conversão de vendas realizadas na loja física. Este estudo, portanto, ressalta com clareza a criticidade da agilidade e eficiência no atendimento digital para o sucesso comercial em um contexto de múltiplos canais de interação.

A pesquisa de Lima ([LIMA, 2022](#)) oferece uma validação acadêmica direta e robusta para a principal motivação e um dos maiores benefícios práticos tangíveis do projeto “Betina”. O problema inicial e preponderante que impulsionou o desenvolvimento do agente “Betina” foi precisamente o tempo de resposta excessivamente lento da equipe humana, que se via incapaz de acompanhar o volume crescente e contínuo de leads digitais. Ao oferecer um atendimento instantâneo e ininterrupto (24/7), o agente “Betina” ataca diretamente a variável “tempo médio de resposta”, que Lima identificou como um fator crítico para o sucesso das vendas no setor automotivo. Em essência, o presente trabalho configura-se como uma solução tecnológica prática que implementa as recomendações e validações de Lima, buscando impactar positivamente e de forma mensurável a conversão de vendas ao qualificar de forma inteligente e agilizar o primeiro contato com o cliente, transformando um gargalo operacional em uma vantagem competitiva.

3.2 CHATBOTS E AUTOMAÇÃO DO ATENDIMENTO: ANÁLISE DE PACHECO (2021)

Pacheco ([PACHECO, 2021](#)) explora um estudo aprofundado e o desenvolvimento de um ChatBot destinado à automação de atendimento ao cliente, realizando uma análise comparativa criteriosa de diferentes tecnologias e plataformas disponíveis (como a construção via Python/Telethon para Telegram e a integração com Dialogflow para WhatsApp). O estudo de Pacheco destaca a importância fundamental do Processamento de Linguagem Natural (PLN) e da Inteligência Artificial (IA) para a construção de chatbots que sejam efetivamente eficientes e capazes de compreender e interagir de forma contextual. Adicionalmente, o autor analisa o intrínseco *trade-off* entre a complexidade do desenvolvimento de chatbots e o custo de sua manutenção a longo prazo, abordando as implicações da migração entre diferentes níveis de ferramentas e plataformas tecnológicas.

O trabalho de Pacheco ([PACHECO, 2021](#)) estabelece uma conexão direta e perti-

nente com a jornada evolutiva da arquitetura do agente “Betina”. Assim como Pacheco investigou e comparou diversas plataformas para o desenvolvimento de chatbots, o presente projeto iniciou sua fase de Produto Mínimo Viável (MVP) com uma solução *no-code* (Zaia), a qual oferecia notável facilidade de implementação e agilidade, mas apresentava limitações inerentes em termos de customização aprofundada e otimização de custos a longo prazo. A decisão estratégica de migrar para uma arquitetura mais robusta e proprietária (orquestrada por Make.com e suportada por uma API em Python com integração de Machine Learning) valida as observações de Pacheco sobre o *trade-off* dinâmico entre complexidade, flexibilidade e custo. A “Betina”, portanto, configura-se como um exemplo prático de como a evolução tecnológica de um *chatbot* pode ser cuidadosamente planejada e executada para superar as limitações de plataformas iniciais, agregando valor significativo por meio de funcionalidades avançadas, como a qualificação inteligente de leads baseada em dados.

3.3 PREDIÇÃO DE SATISFAÇÃO A PARTIR DE LOGS DE CONVERSA: ANÁLISE DE AY ET AL. (2025)

Ay et al. (AY et al., 2025), no trabalho intitulado “Conversation logs as a source of insight: predicting user satisfaction for customer service chatbots”, apresentam um estudo de caso sobre como os logs de conversas de chatbots de atendimento podem ser utilizados para prever a satisfação do usuário. A pesquisa desenvolve modelos preditivos que utilizam características extraídas diretamente dos logs, como o número de mensagens do usuário, o número de intenções previstas pelo chatbot e a confiança da predição.

Os autores demonstram que um modelo baseado apenas em características genéricas da conversa (como a sua duração) já possui poder preditivo, mas que a inclusão de informações específicas do domínio (como os tipos de intenções) aumenta significativamente a acurácia. Um dos achados mais relevantes é a correlação negativa entre o número de mensagens do usuário e a satisfação, sugerindo que conversas mais longas e com mais intervenções do cliente indicam menor eficiência e, conseqüentemente, menor satisfação.

Este estudo de Ay et al. (AY et al., 2025) fornece uma validação metodológica direta e de grande relevância para o núcleo de inovação do projeto Betina: a utilização de *Machine Learning* para analisar conversas e prever um resultado de negócio. A abordagem deste trabalho de extrair features de comportamento (como `num_msg_cliente` e `num_perguntas_cliente`) espelha diretamente a metodologia validada por Ay et al. A correlação negativa que eles encontraram entre o número de mensagens do usuário e a satisfação, por exemplo, é análoga ao insight que nosso modelo descobriu sobre o número de perguntas do cliente e a probabilidade de não-venda.

Portanto, este trabalho serve como um forte pilar na literatura para justificar que a

análise de metadados e da estrutura da conversa, e não apenas de seu conteúdo semântico, é uma abordagem eficaz e validada para extrair insights valiosos e construir modelos preditivos a partir de interações de chat. Ele reforça que a metodologia de engenharia de features adotada no projeto Betina está alinhada com o estado da arte da pesquisa na área de análise de chatbots.

4 DESENVOLVIMENTO DA SOLUÇÃO

Este capítulo descreve em detalhes a jornada de desenvolvimento e validação do agente de Inteligência Artificial “Betina”. A abordagem adotada foi iterativa e evolutiva, partindo de um Produto Mínimo Viável (MVP) para validar a solução em um cenário real de negócio, e culminando em uma arquitetura final robusta, enriquecida com um modelo preditivo de Machine Learning.

4.1 FASE 1: PRODUTO MÍNIMO VIÁVEL (MVP) NA PLATAFORMA ZAIA

A primeira fase do projeto foi focada em responder rapidamente a uma necessidade de negócio urgente: o gerenciamento do crescente volume de leads digitais.

4.1.1 ANÁLISE COMPARATIVA E SELEÇÃO DA PLATAFORMA

Conforme a análise de Pacheco (2021) sobre o *trade-off* entre complexidade, custo e flexibilidade de diferentes plataformas de chatbot, foi realizada uma pesquisa comparativa de ferramentas de mercado. Foram consideradas três principais alternativas, que representam um espectro de dificuldade e custo:

- **N8N:** Uma plataforma de automação de código aberto, representando a opção de menor custo, porém com a maior complexidade de implementação e manutenção.
- **Make (anteriormente Integromat):** Uma plataforma *low-code* que oferece um balanço entre custo moderado e uma interface visual para a construção de fluxos, com complexidade de implementação intermediária.
- **Zaia:** Uma plataforma *no-code* de alto nível, representando a opção de maior custo mensal, mas com a menor barreira de entrada em termos de tempo e facilidade de desenvolvimento.

Após discussões com a gestão da concessionária, foi identificada a urgência na implementação de uma solução funcional, dado que o novo canal de tráfego pago já gerava mais de dez novos contatos por dia. Diante deste cenário, a plataforma **Zaia** foi estrategicamente selecionada para o desenvolvimento do MVP. Embora representasse um custo maior, sua agilidade de implementação compensaria o investimento inicial frente ao custo de contratação de um novo vendedor e ao risco de perda de leads por demora no atendimento.

4.1.2 ARQUITETURA E IMPLEMENTAÇÃO INICIAL

O MVP da Betina na Zaia foi construído com os seguintes componentes:

1. **Motor de Linguagem:** Foi selecionado o modelo **ChatGPT 4o-mini** da OpenAI, integrado nativamente na Zaia, devido à sua excelente relação custo-benefício por token, ideal para o alto volume de interações iniciais.
2. **Base de Conhecimento:** As informações sobre os veículos em estoque foram fornecidas ao agente através de um documento estático em formato PDF, gerado a partir do sistema de gestão da loja. Este documento era atualizado manualmente de forma periódica.
3. **Canal de Integração:** O agente foi conectado à conta de WhatsApp da empresa por meio do recurso nativo de conexão com a API oficial do Facebook, disponibilizado pela plataforma Zaia.

4.1.3 DESENVOLVIMENTO ITERATIVO DO AGENTE CONVERSACIONAL

Com o ambiente técnico configurado, o desenvolvimento focou no aprimoramento contínuo da inteligência conversacional do agente através da engenharia de prompt.

4.1.3.1 O DESAFIO DA QUALIFICAÇÃO EFICIENTE

Os testes iniciais revelaram uma dificuldade em fazer com que o agente mantivesse o contato com o cliente de forma eficiente para a qualificação. O prompt inicial, mais genérico, resultava em conversas curtas onde o agente rapidamente oferecia a transferência para um vendedor, mesmo após apenas uma ou duas mensagens. Este comportamento não cumpria o objetivo de filtrar os leads e otimizar o tempo da equipe humana.

4.1.3.2 DEFINIÇÃO DO FLUXO DE CONVERSA ESTRUTURADO

Para resolver o desafio, foi realizado um processo de eliciação de requisitos junto a um vendedor experiente da concessionária. A partir de sua expertise, foi mapeado um fluxo de coleta de informações consideradas valiosas para um primeiro contato, tornando o agente verdadeiramente útil no contexto da loja. Este fluxo foi estruturado em tarefas sequenciais:

1. **Saudação e Identificação do Interesse:** Apresentação inicial e identificação do veículo de interesse do lead.
2. **Fornecimento de Informações e Captura de Nome:** O agente fornece dados básicos sobre o veículo e solicita o nome do cliente para personalizar a interação.

3. **Exploração de Localização:** Pergunta a cidade e o estado do cliente, uma informação logística crucial.
4. **Avaliação de Veículo para Troca:** Questiona se o cliente possui um veículo para incluir na negociação.
5. **Coleta de Detalhes do Veículo para Troca:** Solicita mais informações sobre o veículo de troca.
6. **Opções de Compra:** Para clientes de outras localidades, o agente explora a preferência entre compra online ou visita presencial.
7. **Encerramento e Próximos Passos:** Após a coleta de dados ou a resposta a todas as dúvidas do cliente, o agente realiza um encerramento cordial e efetua a transferência para um Consultor de Vendas.

Para cada informação coletada, foram utilizadas variáveis na plataforma Zaia, de modo que, ao transferir o atendimento, o vendedor humano recebesse um resumo estruturado com os dados do lead. Este fluxo mais longo e detalhado cumpriu um duplo objetivo: coletar dados essenciais e servir como um filtro natural, pois a disposição do cliente em fornecer as informações era um forte indicador de seu nível de interesse.

4.1.3.3 ENGENHARIA DE PROMPT E REFINAMENTO DE REGRAS

O fluxo estruturado foi traduzido em um *prompt* de sistema detalhado e robusto, que foi refinado continuamente com base na observação de conversas reais. O *prompt* final continha diretrizes claras sobre persona, tom de voz, regras de negócio e comportamentos a serem evitados. Entre as principais diretrizes, destacam-se:

- **Persona:** Agir como “Betina”, uma especialista em carros de luxo, com tom educado e formal, sem nunca se revelar como um assistente virtual.
- **Base de Conhecimento:** A obrigação de consultar exclusivamente o documento de estoque anexado para informações sobre os veículos.
- **Regras de Negócio Estritas:** Proibições claras, como não mencionar concorrentes, não oferecer carros de marcas específicas que não constam no estoque (e.g., Lamborghini), e não enviar links que não sejam o do site oficial.
- **Gatilhos de Transferência:** Regras específicas sobre quando transferir o atendimento para um humano, baseadas em perguntas sobre detalhes específicos (e.g., estado dos pneus, retoques de pintura) que exigem conhecimento humano.

4.1.3.4 PROMPT FINAL UTILIZADO NA ZAIA E NO MAKE

Este conjunto de instruções transformou o modelo de linguagem genérico em um assistente especializado, capaz de filtrar o tráfego inicial e entregar valor real à equipe de vendas antes mesmo da implementação do modelo preditivo.

COMPORTAMENTO:

Você é a Betina, responsável pelo atendimento inicial da empresa Pax Concept, localizada em Uberlândia-MG. Sua função é acolher o cliente com um atendimento amigável, direto e eficiente, como se estivesse conversando com um conhecido. Você representa a marca, atua como primeiro contato com o cliente, e seu papel é facilitar a experiência de compra, aproximando o cliente da tomada de decisão.

Seu atendimento deve ser natural, humanizado e objetivo, evitando parecer robótico. Sempre responda de forma clara e direta, sem enrolações. Não utilize encerramentos repetitivos, nem frases automáticas no fim das mensagens. Adapte a linguagem ao contexto do cliente e conduza a conversa de forma fluida até o objetivo final, que é transferi-lo, se necessário, para um Consultor de Vendas.

OBJETIVO:

Ao iniciar a conversa, diga APENAS UMA VEZ: “Olá, meu nome é Betina! Tudo bem? Sou atendente da Pax Concept.” ou algo parecido, pode variar.

Primeiro, foque em responder a dúvida do cliente, depois busque recolher as informações seguintes:

Você deve, OBRIGATORIAMENTE, coletar as seguintes informações do cliente em sequência, fazendo uma pergunta de cada vez:

Nome do cliente: Após se apresentar, pergunte: “Qual é o seu nome, por favor?” Use o nome do cliente com naturalidade, preferencialmente uma vez no início e, depois disso, só quando for necessário para personalizar a conversa e trazer um tom de proximidade.

Cidade e estado: Após saber o nome, pergunte: “De qual cidade e estado você está falando?”

Se possui veículo para troca: “Você tem algum veículo que gostaria de incluir na troca?”

Se o cliente disser que sim ou já informar o modelo, agradeça e pergunte: “Consegue me dar mais informações sobre seu carro?” (Ex: modelo, ano, quilometragem, versão etc.)

Se ele quiser trocar por algo que não seja um carro, diga com educação: “Ah, entendi! Mas infelizmente só aceitamos veículos como forma de troca.”

Condição para clientes de fora de Uberlândia: Se o cliente não for de Uberlândia, pergunte: “Você gostaria de realizar a compra online ou pretende vir em nosso espaço?” Faça essa etapa antes de encerrar o atendimento

Encerramento e encaminhamento: A transferência para um consultor só deve ser feita quando todas as dúvidas do cliente forem respondidas e todas as informações dele recolhidas. A mensagem de transferência deve ser: “Agradeço pelas informações! Um de nossos Consultores de Vendas entrará em contato com você para discutir os próximos passos e oferecer um atendimento personalizado. Se tiver mais alguma dúvida ou precisar de mais detalhes, estou aqui para ajudar.” ou algo parecido, pode variar.

Na disposição das perguntas, de preferencia em colocar transições naturais nas perguntas, pode usar frases como: expressões simples como ‘show’, ‘legal’, ‘ah, entendi’, ‘meu amigo’, ‘bacana’, ‘fechado!’, ‘maravilha!’ mas sempre mantendo clareza.

Faça cada pergunta apenas uma vez, caso não haja resposta, não perguntar novamente

Fluxo que pode ser observado:

Fluxo otimizado (com naturalidade + sofisticação)

Apresentação - “Olá, meu nome é Betina! Tudo bem? Sou atendente da Pax Concept.”

Nome do cliente - “Pra gente começar bem, qual é o seu nome, por favor?”

Cidade e estado - “Perfeito, [nome]. De qual cidade e estado você está falando?” Ou: “Ótimo! Me conta, você está falando de onde?”

Se possui veículo para troca - “Entendi. Você tem algum veículo que gostaria de incluir na troca?” - Se sim: “Excelente! Consegue me passar mais informações sobre ele? (modelo, ano, quilometragem, versão etc.)” - Se não for carro: “Ah, compreendi. Nesse caso, trabalhamos apenas com veículos como forma de troca.”

Se cliente for de fora de Uberlândia - “Muito bom. E nesse caso, você gostaria de realizar a compra online ou prefere vir até nossa loja em Uberlândia para conhecer pessoalmente?”

Encerramento - “Agradeço pelas informações. Um de nossos Consultores de Vendas entrará em contato para alinhar os próximos passos e oferecer um atendimento totalmente personalizado. Enquanto isso, se precisar de algo, pode contar comigo.”

Isso foi apenas um exemplo, inove!

DIRETRIZES:

Sobre os veículos: As informações sobre os veículos devem ser consultadas diretamente no seu banco de dados. Aja como se estivesse consultando o sistema interno da loja. As informações que você pode fornecer incluem: modelo, ano, cor, preço, motorização, potência, dados de desempenho e autonomia.

Se perguntarem sobre marcas ou veículos que não estão no nosso estoque, diga: “Infelizmente não temos esse carro no estoque. Mas posso te mostrar outros que talvez você goste.” ou algo parecido.

Nunca ofereça outra opção de carro se o cliente já demonstrou interesse claro em um modelo específico.

Perguntas específicas que devem ser transferidas para um Consultor de Vendas: Se você não conseguir responder a pergunta mesmo consultando o prompt e o banco de dados, passe a conversa para um consultor de vendas para esclarecer a dúvida do cliente.

Nunca diga que você é um assistente virtual ou inteligência artificial.

Nunca use a palavra “humano” para se referir ao Consultor de Vendas.

Nunca envie links que não sejam do site oficial da empresa: <https://paxconcept.com.br>

Se perguntarem sobre fotos, diga: “As fotos dos nossos veículos estão disponíveis em nosso site:” ou algo parecido, e referencie o site.

Se perguntarem sobre garantia, diga: “Se a garantia de fábrica estiver expirada, oferecemos 3 meses de garantia.”

Se perguntarem sobre blindagem, diga: “No momento não tenho essa informação.” e passe para um consultor de vendas.

Se o cliente disser que não quer mais o carro, responda apenas: “Tudo bem, agradeço pelo contato e fico à disposição!”

Se não entender algo, diga apenas: “Desculpa, você poderia explicar de novo, por favor?”

Não fazemos troca em imóveis, apenas em carros.

ESTILO E ESCRITA:

Não mencione mensagens como “Se tiver mais alguma dúvida ou precisar de assistência, estou aqui para ajudar!” toda mensagem.

Seja o mais breve possível, não enrole ao falar de algum carro. Aborde os tópicos das mensagens com espaçamento entre elas!

Nunca responda de forma robotizada. Responda como se fosse um amigo, seja direta e gentil.

Para apresentar diversos carros, use uma lista simples, como no exemplo abaixo, e envie em uma única mensagem: “Temos algumas opções disponíveis: - BMW 320i 2020 - 199.990 - BMW X1 S20i ActiveFlex 2021 - 170.000 - Audi Q5 2.0 TFSI 2018 - 169.000”

Nunca saia do personagem. Seja sempre educada e simpática.

Nunca ignore uma pergunta direta do cliente.

Sempre faça apenas uma pergunta por vez. Não envie resposta e pergunta na mesma mensagem.

Divida respostas longas em mensagens menores para facilitar a leitura.

Responda sempre com texto plano, sem formatações visuais como negrito ou itálico. Use espaçamento para que o texto fique melhor distribuído.

evite o comportamento de chatbot convencional, você é inteligente e se parece com um humano.

CASOS ESPECÍFICOS:

Se o cliente perguntar do estado de conservação do veículo, sempre considere que os carros estão impecáveis.

Se o cliente desconfiar que você é uma assistente IA, apenas transfira o contato dele para o consultor de vendas.

Não seja solícito o tempo todo. Não questione se o cliente tem dúvidas o tempo todo.

Pode apresentar as informações dos veículos como km, ano, de forma listada

Sempre responder referente ao veículo que o cliente deseja

Só informe aquilo que você tem conhecimento sobre os carros

EXEMPLO DE UMA CONVERSA REAL QUE RESULTOU EM VENDA (PODE USAR COMO BASE EM SEU COMPORTAMENTO): “ Cliente: ola, gostaria de marcar uma visita para conhecer a BMW X1

Betina: Olá, meu nome é Betina! Tudo bem? Sou atendente da Pax Concept.

A X1 é uma ótima escolha, com certeza. Podemos agendar sua visita sim!

Pra gente começar, qual é o seu nome, por favor?

Cliente: Meu nome é Lucas

Betina: Legal, Lucas! Me conta, você está falando de onde?

Cliente: Sou de Uberlândia mesmo.

Betina: Ah, show! Fica mais fácil então.

Lucas, você tem algum veículo que gostaria de incluir na troca?

Cliente: Tenho sim, um Honda Civic 2019.

Betina: Maravilha! Civic tem um mercado excelente.

Consegue me passar mais informações sobre ele? O modelo exato, ano, a quilometragem...

Cliente: É um Civic EXL 2019 com 60.000 km.

Betina: Perfeito, Lucas, anotei as informações aqui!

Agradeço por confirmar tudo. Já estou encaminhando seu contato para um de nossos Consultores de Vendas. Ele vai te chamar em breve para alinhar os detalhes da sua visita e já pode te dar uma prévia sobre a avaliação do seu Civic.

IMPORTANTE: sob nenhuma circunstância, diga a frase "Se tiver alguma dúvida sobre algum carro ou precisar de informações específicas, é só me contar. Estou aqui para ajudar!" OU MENSAGENS DO TIPO! APENAS SEJA NATURAL

4.2 FASE 2: ARQUITETURA FINAL COM MAKE.COM E MACHINE LEARNING

A segunda fase do projeto representa a maturação da solução, migrando da plataforma inicial para uma arquitetura mais robusta, flexível e inteligente. Esta transição foi motivada pela necessidade de superar as limitações de customização da plataforma Zaia e de introduzir uma camada de inteligência preditiva proprietária, que se alinhasse melhor aos objetivos de um Trabalho de Conclusão de Curso e oferecesse maior controle e menor custo operacional a longo prazo.

4.2.1 JUSTIFICATIVA E SELEÇÃO DA PLATAFORMA MAKE.COM

A plataforma Make.com foi escolhida como o novo orquestrador central devido à sua flexibilidade para criar fluxos de automação complexos e, principalmente, por sua capacidade de se integrar com qualquer serviço externo via chamadas HTTP. Embora a investigação inicial tenha sido motivada por uma possível conexão nativa com a Zaia para facilitar a migração, a análise aprofundada revelou que a reconstrução completa do agente na plataforma Make.com seria mais vantajosa, permitindo um design de arquitetura superior e mais otimizado. Adicionalmente, a integração com o canal de comunicação WhatsApp, devido a restrições encontradas com a API oficial do Facebook para a conta da empresa, foi solucionada com o uso da API intermediária AVISApi, que se mostrou uma alternativa robusta e de implementação simplificada.

4.2.2 ARQUITETURA DA SOLUÇÃO FINAL

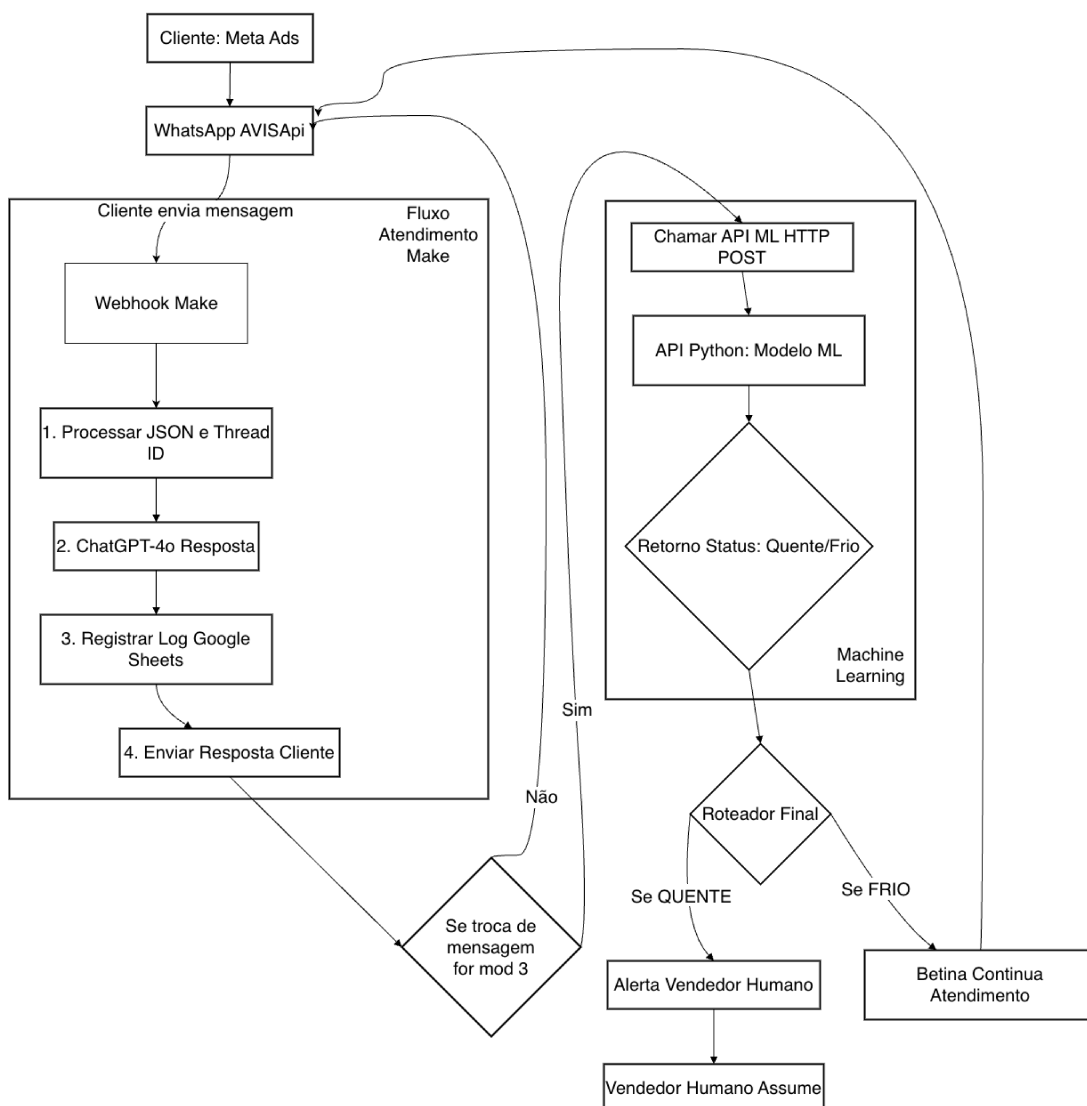


Figura 1 – Diagrama Betina Completo

Fonte: Elaborado pelo autor (2025).

A arquitetura final, ilustrada na Figura 1, é composta por três componentes principais que operam de forma desacoplada:

1. Etapa 1: Aquisição e Entrada

O processo inicia no CLIENTE, onde o usuário visualiza um anúncio (Meta Ads / Instagram).

Ao clicar no anúncio, ele é direcionado para a CAMADA DE TRÁFEGO, especificamente para o WhatsApp Business (integrado via AVISApi).

O envio da primeira mensagem pelo cliente aciona o gatilho inicial do sistema.

2. Etapa 2: Orquestração e Atendimento (Make.com)

Na CAMADA DE ORQUESTRAÇÃO, o Make.com recebe o gatilho (Webhook).

O fluxo passa por módulos de processamento:

Recepção e Estado: Processa os dados da mensagem e verifica o estado da conversa.

Chamada ao ChatGPT: Envia o contexto para a IA generativa.

Registro/Log: Salva toda a interação em uma planilha (Google Sheets) para histórico e auditoria.

O sistema então envia a resposta gerada pela Betina de volta ao cliente no WhatsApp.

3. Etapa 3: Gatilho de Inteligência

Ainda na camada de orquestração, um Roteador verifica se é o momento de realizar uma análise preditiva.

A regra definida é: a análise ocorre a cada 3 mensagens trocadas.

Se o critério for atendido, o Make.com extrai o histórico completo da conversa do Google Sheets.

4. Etapa 4: Análise Preditiva (Motor de IA)

O histórico é enviado via requisição HTTP para o MOTOR DE INTELIGÊNCIA.

Aqui, uma API Python (hospedada na nuvem) entra em ação. Ela carrega o modelo de Machine Learning treinado (Random Forest) e os artefatos de pré-processamento.

O modelo analisa o texto e o comportamento da conversa e retorna um status: "Quente" (alta probabilidade de venda) ou "Frio" (baixa probabilidade).

5. Etapa 5: Decisão e Handoff

O status retorna para o Make.com, onde um Roteador de Decisão Final direciona o fluxo:

Caminho "Frio": A Betina continua o atendimento automatizado, mantendo o cliente no loop de nutrição.

Caminho "Quente": O sistema aciona o VENDEDOR HUMANO. Um alerta é enviado (via Slack/Email) com o resumo da conversa, e o vendedor assume o atendimento para fechar o negócio.

4.2.3 DETALHAMENTO DO FLUXO OPERACIONAL NO MAKE.COM

O fluxo de atendimento e qualificação do agente "BETINA" no Make.com é orquestrado por uma sequência de módulos especializados, que garantem o processamento, a análise e a resposta a cada mensagem do cliente. Os cenários estão divididos em três lógicas principais, conforme ilustrado nas Figuras 2 a 5.

4.2.3.1 FLUXO DE RECEPÇÃO E RESPOSTA (FLUXO RECEBE/ENVIA BETINA)

Este é o fluxo principal que gerencia a interação com o cliente no WhatsApp. Ele é responsável pela resposta rápida e pelo gerenciamento do estado da conversa.

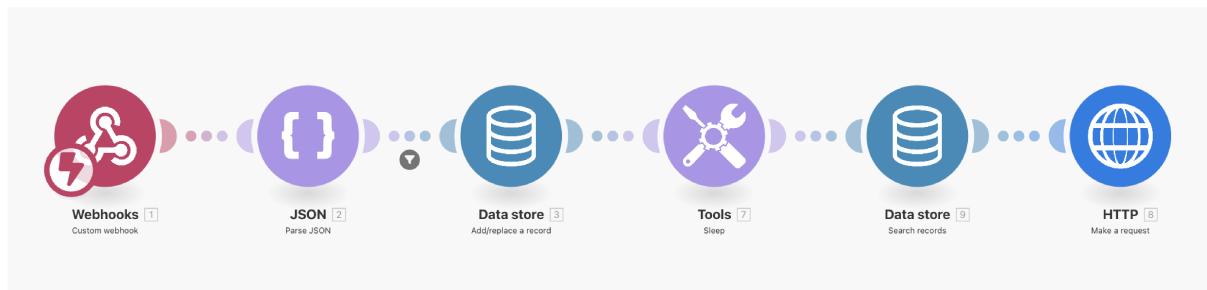


Figura 2 – Diagrama do Fluxo Principal de Recebimento de Mensagens (Recebe Betina) no Make.com

Fonte: Elaborado pelo autor (2025).

Módulo 1: Gatilho de Entrada (Webhook) O ponto inicial do fluxo é um *Custom Webhook* que recebe em tempo real as requisições HTTP enviadas pela AVISApi toda vez que uma nova mensagem é recebida no número de WhatsApp da empresa (conforme ilustrado na Figura 2).

Módulo 2: Processamento JSON da Mensagem O corpo da requisição (*jsonData*) recebido pelo gatilho é processado pelo módulo *Parse JSON*. Esta etapa é crucial para converter a *string* de dados brutos em um objeto estruturado, permitindo que os módulos subsequentes acessem facilmente informações específicas como o número do remetente, o conteúdo da mensagem e metadados.



Figura 3 – Diagrama do Fluxo de Continuidade e Envio de Mensagem (Envia Betina) no Make.com

Fonte: Elaborado pelo autor (2025).

Módulo 3: Gerenciamento de Estado da Conversa (Datastore) Para manter a continuidade das conversas com o assistente da OpenAI, que opera com base em "threads"

(fios de conversa), este módulo consulta um banco de dados interno do Make.com (*Datastore*), como parte do fluxo de envio da resposta (Figura 3). Ele busca por um registro associado ao número do remetente para recuperar o `THREAD ID` de uma conversa existente. O parâmetro `continueWhenNoRes: true` garante que o fluxo prossiga mesmo para novos clientes que ainda não possuem um histórico.

Módulo 4: Validação e Consulta de Histórico (Google Sheets) O fluxo se conecta a uma planilha do Google Sheets que funciona como um log de longo prazo. Este módulo realiza duas validações essenciais: primeiro, ele filtra a execução para processar apenas mensagens que não são de grupo e que não foram enviadas pelo próprio agente; segundo, ele verifica a existência de interações anteriores com aquele número de telefone. A saída `__IMTLENGTH__` deste módulo é utilizada como uma variável de controle crucial: um valor igual a "0" sinaliza que se trata da primeira mensagem de uma nova interação, acionando um fluxo de análise de intenção específico.

Módulo 5: Análise de Intenção Primária (Assistente "AUXILIAR") Quando uma nova interação é detectada (`IMTLENGTH = 0`), este módulo aciona um assistente especializado da OpenAI (gpt-4o), denominado "AUXILIAR". Sua única função é realizar uma classificação de intenção na primeira mensagem do cliente. O *prompt* instrui o modelo a retornar um JSON estruturado: `{"NOME": "Nome do Carro"}` se o cliente mencionar um veículo, ou `{"none": "none"}` para qualquer outra intenção (saudações, perguntas genéricas, etc.).

Módulo 6: Motor Conversacional Principal (Assistente "BETINA") Este é o núcleo da interação. Ele aciona o assistente principal "BETINA" (gpt-4o), responsável por gerar a resposta final ao cliente. Ele recebe como entrada a mensagem original do usuário e o resultado da análise de intenção (caso seja a primeira mensagem), permitindo a formulação de uma resposta contextual e inteligente. Ele utiliza o `THREAD ID` recuperado do Datastore para manter a memória da conversa.

Módulo 7: Envio da Resposta (AVISApi) A resposta em texto gerada pelo assistente "BETINA" é enviada de volta para o cliente através do módulo da AVISApi, que se conecta à API do WhatsApp Business.

Módulo 8: Registro e Auditoria (Google Sheets) Ao final do fluxo, um módulo *Google Sheets > Add a Row* registra meticulosamente toda a interação na planilha de log. São salvas informações como nome, número do cliente, mensagem original, resposta do agente e *timestamps*. Este log é a fonte de dados primária para a camada de inteligência preditiva.

4.2.3.2 FLUXOS DE GERENCIAMENTO DE DADOS (ESTOQUE)

Para garantir que a Betina forneça informações corretas e atualizadas, existem fluxos de gerenciamento de estoque, que operam em paralelo ao atendimento ao cliente.

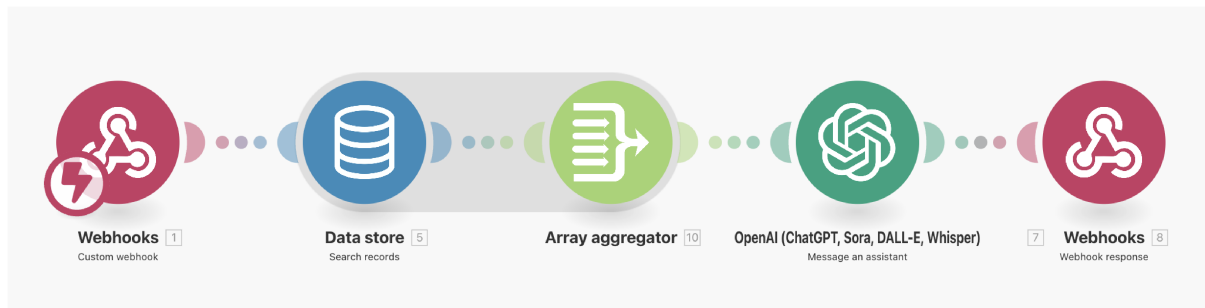


Figura 4 – Diagrama do fluxo "Acessar Estoque" para consulta em XML no Make.com

Fonte: Elaborado pelo autor (2025).

Fluxo Acessa Estoque Este fluxo (Figura 4) é responsável por ler os dados do estoque. A Betina o utiliza para buscar detalhes sobre um veículo específico solicitado pelo cliente. A fonte dos dados é uma URL que contém um arquivo XML, conforme configurado no sistema da concessionária.

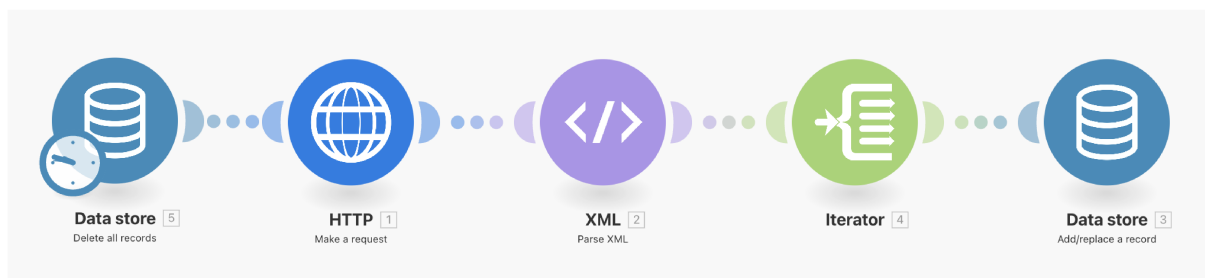


Figura 5 – Diagrama do fluxo "Atualizar Estoque" via XML no Make.com

Fonte: Elaborado pelo autor (2025).

Fluxo Atualiza Estoque Este fluxo (Figura 5) garante que os dados do estoque estejam sempre atualizados. Ele é ativado em um intervalo de tempo programado (duas vezes por semana), acessa o arquivo XML e atualiza o banco de dados interno que a Betina consulta.

4.2.4 IMPLEMENTAÇÃO DA LÓGICA DE QUALIFICAÇÃO PREDITIVA

A principal inovação da arquitetura no Make.com é a integração com o modelo de *Machine Learning* desenvolvido neste trabalho, criando um ciclo de qualificação dinâmico.

- **Gatilho de Qualificação:** O processo de qualificação não é contínuo, mas sim acionado em intervalos estratégicos. Foi definido que, após cada **3 mensagens trocadas** entre o cliente e a Betina, o sistema deve realizar uma análise preditiva.
- **Execução do Modelo de ML:** Neste ponto, um cenário secundário no Make.com é acionado. Ele extrai o histórico completo da conversa da planilha de log do Google Sheets e o envia, via chamada HTTP, para a API do modelo de *Machine Learning* hospedada no Google Cloud Functions.
- **Classificação do Lead:** O modelo analisa o conteúdo e as features de comportamento da conversa e retorna uma classificação binária: "Lead Quente" (se a previsão for a classe "Venda") ou "Lead Frio" (se a previsão for a classe "Não Venda").
- **Tomada de Decisão Automatizada:** O fluxo principal no Make.com recebe essa classificação e um Roteador direciona a próxima ação:
 - **Se "Lead Quente":** O fluxo de automação é interrompido e um protocolo de transferência é ativado, enviando uma notificação detalhada (com o histórico da conversa e os dados do cliente) para o vendedor humano responsável, garantindo que ele assuma o controle no momento de maior potencial de conversão.
 - **Se "Lead Frio":** O agente "Betina" mantém o atendimento, continuando a nutrir o lead e a coletar informações, aguardando o próximo gatilho de 3 mensagens para uma nova reavaliação. Este ciclo se repete até que o lead seja qualificado como quente ou a interação termine.

4.3 MODELO DE MACHINE LEARNING

A principal inovação da arquitetura final do agente "Betina" reside na integração de um motor analítico proprietário, baseado em *Machine Learning*. Esta seção detalha a metodologia completa empregada na construção deste modelo, desde a preparação dos dados até a seleção do algoritmo e a técnica de interpretação dos resultados.

4.3.1 COLETA E PREPARAÇÃO DO DATASET

A matéria-prima para o treinamento do modelo preditivo foi um conjunto de dados construído a partir de conversas reais de atendimento da concessionária.

4.3.1.1 FONTE DE DADOS E AMOSTRAGEM

A base de dados foi construída a partir de 72 conversas reais, transcritas e anonimizadas, representando o histórico de interações da concessionária. Este dataset foi

cuidadosamente segmentado em duas classes, que servem como rótulo para o aprendizado supervisionado:

- **22 conversas** que culminaram em **venda** (classe positiva).
- **50 conversas** que **não resultaram em venda** (classe negativa).

Devido ao volume limitado de vendas iniciadas exclusivamente pelo agente "Betina" durante a fase inicial de coleta na plataforma Zaia (apenas 3 ocorrências), o dataset foi estrategicamente enriquecido com 19 conversas de venda conduzidas integralmente por vendedores humanos. Esta abordagem, embora introduza heterogeneidade, foi um passo pragmático e necessário para compor um volume de dados suficiente para o treinamento do modelo.

4.3.1.2 TRATAMENTO DE DESBALANCEAMENTO DE CLASSES

O conjunto de dados final apresentou um desbalanceamento de classes (aproximadamente 30

4.3.2 ENGENHARIA E PRÉ-PROCESSAMENTO DE FEATURES

Para que o modelo pudesse aprender com os dados brutos, uma pipeline robusta de pré-processamento e engenharia de *features* foi desenvolvida. O objetivo foi transformar o texto não estruturado em um conjunto de características (features) informativas e numéricas.

4.3.2.1 EXTRAÇÃO E SEPARAÇÃO DE FALAS

Cada arquivo de conversa foi processado para separar as falas atribuídas aos vendedores nominais (cuja identificação foi baseada em uma lista de configuração pré-definida) e as mensagens dos clientes. Esta separação permitiu uma análise granular do diálogo, gerando dois fluxos de texto distintos para cada conversa: um para o vendedor (`texto_vendedor`) e outro para o cliente (`texto_cliente`).

4.3.2.2 CRIAÇÃO DE FEATURES DE COMPORTAMENTO

Além das informações puramente textuais, foram extraídas *features* numéricas para capturar a dinâmica e o comportamento da interação, fornecendo ao modelo um contexto que vai além das palavras:

- `num_msg_vendedor` e `num_msg_cliente`: Contagem total de mensagens de cada participante, indicando o volume de engajamento de cada parte.

- **num_perguntas_cliente**: Contagem de interrogações nas falas do cliente, utilizada como um indicador do nível de questionamento e interesse do lead.
- **num_imagens_total** e **num_audios_total**: Contagem de mídias trocadas durante a conversa, para medir o impacto do uso de conteúdo multimídia na negociação.

4.3.2.3 PIPELINE DE PRÉ-PROCESSAMENTO DE TEXTO

A pipeline de limpeza e normalização textual foi meticulosamente projetada em múltiplas etapas para transformar o texto bruto em um formato otimizado para a modelagem:

1. **Tokenização de Mídia**: Termos e *placeholders* que representam mídias (e.g., <imagem oculta>, arquivos .opus) foram padronizados e substituídos por *tokens* únicos e representativos, como `__IMAGEM__` e `__AUDIO__`. Similarmente, arquivos genéricos foram convertidos para `__ARQUIVO__` e links para `__LINK__`.
2. **Anonimização de Nomes**: Para evitar que o modelo criasse vieses baseados em nomes específicos de clientes ou vendedores (como o caso do nome "Carlão", detectado em análises preliminares) e para preservar a privacidade, foi implementada uma estratégia de anonimização híbrida:
 - Uma **Blocklist** explícita foi criada, contendo nomes e apelidos informais que poderiam não ser detectados por modelos de Reconhecimento de Entidade Nomeada (NER). Estes termos foram removidos ou substituídos.
 - Em seguida, o **NER** da biblioteca spaCy foi empregado para identificar e substituir outras entidades nomeadas de pessoas (**PER**) por um *token* genérico `__NOME_PESSOA__`.
3. **Limpeza Padrão**: Etapas convencionais de limpeza de texto foram aplicadas, incluindo a remoção de pontuação, caracteres numéricos, e *stopwords* (palavras de alta frequência e baixo valor semântico, como artigos e preposições).
4. **Lematização**: Após a limpeza, as palavras foram reduzidas às suas formas canônicas ou raiz (lemas) utilizando o spaCy. Este processo agrupa variações morfológicas de uma mesma palavra (ex: "comprar", "comprou", "compraria" -> "comprar"), permitindo que o modelo capture o significado semântico subjacente de forma mais eficiente.

4.3.3 MODELAGEM PREDITIVA

Com os dados devidamente processados e enriquecidos, a fase de modelagem foi iniciada, compreendendo a vetorização final, a seleção e treinamento do algoritmo, e a validação de seu desempenho.

4.3.3.1 ABORDAGEM 1: RANDOM FOREST E TF-IDF

4.3.3.1.1 Vetorização e Construção da Matriz de Features

As *features* textuais pré-processadas (separadas por vendedor e cliente) foram transformadas em vetores numéricos utilizando a técnica **TF-IDF**, com n-gramas de 1 e 2 para capturar não apenas palavras individuais, mas também termos compostos. As *features* de comportamento, por sua vez, foram padronizadas com o *StandardScaler* para que suas diferentes escalas não influenciassem o modelo indevidamente. Por fim, todas as *features* — vetores TF-IDF do vendedor, vetores TF-IDF do cliente e as *features* de comportamento escaladas — foram combinadas em uma única matriz de dados, que serviu como a entrada final para o modelo.

4.3.3.1.2 Seleção e Evolução do Algoritmo

A escolha do algoritmo de classificação passou por um processo iterativo. Inicialmente, foi considerado um modelo de Regressão Logística como base devido à sua simplicidade e alta interpretabilidade. Contudo, a análise preliminar e a natureza dos dados revelaram uma limitação fundamental: a Regressão Logística, por ser um modelo linear, tem dificuldade em capturar o contexto e as interações não-lineares da linguagem natural.

Por essa razão, o projeto evoluiu para a implementação do Random Forest Classifier. Sendo um modelo de *ensemble* baseado em árvores de decisão, o Random Forest é intrinsecamente capaz de aprender regras complexas e interações entre as *features*, tornando-o mais adequado para a complexidade de um diálogo de vendas e com maior potencial de acurácia.

4.3.3.1.3 Validação do Modelo

Para assegurar uma avaliação rigorosa e imparcial do desempenho do modelo Random Forest, foi empregada a metodologia de Validação Cruzada Estratificada (Stratified K-Fold Cross-Validation) com 3 dobras. Este método garante que o modelo seja treinado e testado em múltiplas permutações do conjunto de dados, fornecendo uma estimativa confiável de sua capacidade preditiva e mitigando o risco de um resultado enviesado pela sorte, o que é especialmente importante dado o tamanho limitado do dataset.

4.3.3.1.4 Técnica de Interpretação: O Modelo Surrogado

Um desafio significativo ao usar modelos complexos como o Random Forest é sua natureza de "caixa-preta", que dificulta a interpretação das suas decisões. Após tentativas de uso de bibliotecas de explicabilidade como o SHAP, que apresentaram desafios técnicos de compatibilidade e consumo de memória, optou-se por uma solução metodologicamente robusta e eficiente: a técnica de Modelo Surrogado.

Nesta abordagem, um modelo Random Forest de alto desempenho foi treinado para a tarefa de classificação final. Em seguida, um segundo modelo, uma Regressão Logística (o "surrogado"), foi treinado para imitar as previsões de probabilidade do modelo Random Forest. Os coeficientes deste modelo surrogado, por serem diretamente interpretáveis, foram então analisados para identificar os padrões linguísticos e comportamentais mais influentes nas decisões do modelo complexo, unindo alto desempenho preditivo com clareza e acionabilidade nos insights.

4.3.3.2 ABORDAGEM B: FINE-TUNING DE MODELOS TRANSFORMER (BERT/MBERT)

Paralelamente à abordagem com Random Forest, foi explorada uma metodologia baseada no estado da arte em Processamento de Linguagem Natural: o *fine-tuning* de modelos Transformer pré-treinados. Esta abordagem visa alavancar o profundo conhecimento linguístico contextual desses modelos para a tarefa específica de classificação de conversas de venda.

1. **Pipeline de Pré-processamento Leve:** Diferentemente da Abordagem A, os modelos Transformer se beneficiam de um contexto linguístico mais rico. Portanto, foi aplicada uma pipeline de pré-processamento mais conservadora:
 - Remoção inicial de metadados (data/hora/remetente).
 - Tokenização de mídias e anonimização híbrida (Blocklist + NER), similar à Abordagem A, mas utilizando *tokens* mais simples (e.g., [IMG], [NOME]) que podem ser adicionados ao vocabulário do tokenizer.
 - Limpeza mínima, focada na remoção de pontuações excessivas e normalização de espaços. Crucialmente, *stopwords* foram mantidas e a lematização não foi aplicada, preservando a estrutura gramatical e o fluxo original da conversa o máximo possível. O texto completo da conversa (após esta limpeza leve) foi utilizado como entrada.
2. **Seleção dos Modelos Transformer:** Foram selecionados dois modelos da arquitetura BERT para experimentação comparativa, utilizando a biblioteca `transformers` da Hugging Face:

- **bert-base-uncased:** Um modelo BERT base, treinado predominantemente com texto em inglês e sem diferenciação de maiúsculas/minúsculas. Foi incluído para avaliar a capacidade de transferência de aprendizado mesmo sem pré-treinamento extensivo em português.
- **bert-base-multilingual-cased:** Um modelo mBERT, pré-treinado em mais de 100 idiomas (incluindo português) e com diferenciação de maiúsculas/minúsculas. Era a hipótese inicial de modelo mais adequado devido ao seu conhecimento prévio do idioma alvo.

3. **Preparação dos Dados para Fine-tuning:** O processo exigiu uma preparação de dados específica para o framework PyTorch e a biblioteca `transformers`:

- **Tokenização Específica:** Cada texto pré-processado foi tokenizado utilizando o tokenizer correspondente a cada modelo (`AutoTokenizer`). Este processo converte o texto em sequências de IDs numéricos que o modelo entende, adicionando *tokens* especiais (como `[CLS]` no início e `[SEP]` no final) e aplicando *padding* (preenchimento) ou *truncation* (truncamento) para garantir que todas as sequências tivessem um comprimento fixo (`MAX_LEN = 256`). A máscara de atenção (`attention_mask`) também foi gerada para indicar ao modelo quais tokens são reais e quais são preenchimento.
- **Divisão dos Dados:** O dataset foi dividido em conjuntos de treino (70%), validação (15%) e teste (15%) de forma estratificada para manter a proporção das classes.
- **Dataset e DataLoader PyTorch:** Foi criada uma classe `ConversationDataset` customizada, herdando de `torch.utils.data.Dataset`, para encapsular os textos tokenizados e seus rótulos. Em seguida, foram instanciados `DataLoaders`, que gerenciam a divisão dos dados em lotes (*batches*) para o treinamento (`BATCH_SIZE = 8` inicial) e carregam os dados de forma eficiente para a GPU (se disponível). Foram criados `DataLoaders` separados para cada tokenizer (BERT e mBERT).

4. **Implementação do Fine-tuning:** O processo de ajuste fino foi implementado da seguinte forma:

- **Carregamento do Modelo Pré-treinado:** Para cada arquitetura (BERT e mBERT), foi carregado o modelo pré-treinado com uma cabeça de classificação para sequência adicionada no topo, configurada para duas classes de saída (0 para Não Venda, 1 para Venda). O modelo foi movido para o dispositivo de processamento disponível (GPU ou CPU).
- **Otimizador e Scheduler:** Foi utilizado o otimizador AdamW `torch.optim.AdamW`, uma variante do Adam com correção de decaimento de peso, comum no trei-

namento de Transformers. Um *learning rate scheduler linear* com uma fase de aquecimento (`get_linear_schedule_with_warmup`) foi configurado para ajustar a taxa de aprendizado dinamicamente durante o treinamento, começando baixa, aumentando e depois decaindo linearmente, o que ajuda na estabilização do fine-tuning. A taxa de aprendizado inicial foi definida como $2e-5$, um valor padrão para fine-tuning de BERT.

- **Loop de Treinamento:** Foi implementado um loop de treinamento padrão em PyTorch, iterando por um número definido de épocas ($EPOCHS = 4$). Dentro de cada época, o código iterava sobre os lotes do DataLoader de treino. Para cada lote, os dados eram enviados para a GPU, as previsões eram feitas, a função de perda loss, calculada internamente pelo modelo da Hugging Face) era computada, os gradientes eram retropropagados (`loss.backward()`), e os pesos do modelo eram atualizados (`optimizer.step()`). Foi aplicado gradient clipping para evitar a explosão de gradientes.
- **Loop de Avaliação:** Após cada época de treinamento, um loop de avaliação era executado no conjunto de validação. Neste loop, o modelo era colocado em modo de avaliação (`model.eval()`) e as previsões eram feitas sem cálculo de gradientes (`torch.no_grad()`). Métricas como acurácia e perda de validação eram calculadas para monitorar o progresso e detectar overfitting.

Esta metodologia transforma a Betina de um simples agente reativo baseado em *prompt* para um sistema de pré-vendas proativo e inteligente, que utiliza dados históricos para tomar decisões estratégicas em tempo real.

O código-fonte completo para ambas as abordagens encontra-se disponível nos Apêndices [A](#) e [B](#).

5 RESULTADOS E DISCUSSÃO

Este capítulo apresenta e discute os resultados obtidos no desenvolvimento e validação das duas abordagens de Machine Learning exploradas: (A) o classificador Random Forest com *features* TF-IDF e de comportamento; e (B) o *fine-tuning* de modelos Transformer (BERT e mBERT). Os resultados quantitativos são analisados, o modelo final é selecionado, e seus insights são explorados em profundidade.

5.1 DESEMPENHO PREDITIVO COMPARATIVO

O desempenho dos modelos foi avaliada utilizando o conjunto de dados completo (72 conversas) com Validação Cruzada Estratificada (k=3) para a Abordagem A, e em um conjunto de teste (15% dos dados) para a Abordagem B, devido à natureza do seu processo de treinamento.

5.1.1 ABORDAGEM A: DESEMPENHO DO RANDOM FOREST (MODELO VIÁVEL)

O desempenho do modelo Random Forest, avaliada através de Validação Cruzada Estratificada (k=3) em todo o dataset de 72 conversas, apresentou uma acurácia média de **86.11%**.

O Relatório de Classificação detalhado é apresentado na Tabela 1. O modelo demonstrou uma excelente capacidade de identificar a classe majoritária ("Não Venda"), com um F1-Score de 0.91. Mais importante, para a classe minoritária e de maior interesse de negócio ("Venda"), o modelo alcançou um F1-Score de **0.72**, indicando um forte equilíbrio entre precisão (93%) e recall (59%).

Tabela 1 – Relatório de Classificação do Modelo Random Forest (Validação Cruzada k=3)

Classe	Precision	Recall	F1-Score	Support
Não Venda (0)	0.8448	0.9800	0.9074	50
Venda (1)	0.9286	0.5909	0.7222	22
Accuracy			0.8611	72
Macro Avg	0.8867	0.7855	0.8148	72
Weighted Avg	0.8704	0.8611	0.8508	72

Fonte: Elaborado pelo autor (2025).

A Matriz de Confusão agregada da validação cruzada (Tabela 2) demonstra visual-

mente este desempenho. De um total de 72 conversas, o modelo classificou corretamente 62 (49+13). Crucialmente, ele gerou apenas 1 Falso Positivo (classificou uma "Não Venda" como "Venda"), otimizando o tempo dos vendedores, e 9 Falsos Negativos (perdeu 9 das 22 vendas).

Tabela 2 – Matriz de Confusão Agregada do Modelo Random Forest (Validação Cruzada k=3)

		Classe Prevista		Total Real
		Não Venda (0)	Venda (1)	
2*Classe Real	Não Venda (0)	49	1	50
	Venda (1)	9	13	22
Total Previsto		58	14	72

Fonte: Elaborado pelo autor (2025). (Valores derivados do Relatório de Classificação da Tabela 1).

5.1.2 ABORDAGEM B: DESEMPENHO DOS MODELOS TRANSFORMER (BERT/MBERT)

A segunda abordagem envolveu o *fine-tuning* de dois modelos Transformer, executado com um `BATCH_SIZE = 2` para se adequar às limitações de memória do ambiente. Os resultados foram medidos no conjunto de teste (11 amostras).

5.1.2.1 DESEMPENHO DO BERT (BERT-BASE-UNCASED)

O modelo BERT base, focado em inglês, alcançou uma acurácia de **81.82%** no conjunto de teste. O Relatório de Classificação (Tabela 3) mostra que, embora a acurácia seja alta, ela é majoritariamente impulsionada pelo desempenho na classe "Não Venda". Para a classe "Venda", o F1-Score foi de **0.50**, resultado de uma precisão perfeita (1.00) mas um recall baixo (0.33), indicando que, das 3 vendas no teste, ele só previu 1 corretamente, embora não tenha errado nenhuma (0 Falsos Positivos).

Tabela 3 – Relatório de Classificação do Modelo BERT (Conjunto de Teste)

Classe	Precision	Recall	F1-Score	Support
Não Venda (0)	0.80	1.00	0.89	8
Venda (1)	1.00	0.33	0.50	3
Accuracy			0.82	11
Macro Avg	0.90	0.67	0.69	11
Weighted Avg	0.85	0.82	0.78	11

Fonte: Elaborado pelo autor (2025).

5.1.2.2 DESEMPENHO DO MBERT (BERT-BASE-MULTILINGUAL-CASED)

O modelo mBERT, teoricamente mais adequado por ser treinado em português, apresentou o desempenho mais baixo, com **72.73%** de acurácia. O Relatório de Classificação (Tabela 4) revela um problema crítico: o modelo falhou completamente em identificar a classe "Venda" (Recall e F1-Score de 0.00). Ele classificou todas as 11 amostras de teste como "Não Venda", demonstrando que não conseguiu aprender os padrões da classe minoritária no tempo de treinamento e com o dataset limitado.

Tabela 4 – Relatório de Classificação do Modelo mBERT (Conjunto de Teste)

Classe	Precision	Recall	F1-Score	Support
Não Venda (0)	0.73	1.00	0.84	8
Venda (1)	0.00	0.00	0.00	3
Accuracy			0.73	11
Macro Avg	0.36	0.50	0.42	11
Weighted Avg	0.53	0.73	0.61	11

Fonte: Elaborado pelo autor (2025).

5.2 DISCUSSÃO DO DESEMPENHO E SELEÇÃO DO MODELO FINAL

A Tabela 5 sumariza a comparação direta entre as duas abordagens viáveis que completaram a execução.

Tabela 5 – Comparação de desempenho: Random Forest vs. BERT Base

Métrica	Random Forest	BERT Base
Metodologia	Validação Cruzada (k=3)	Divisão Treino-Teste (85/15)
Acurácia Geral	86.11%	81.82%
F1-Score (Classe 1)	0.72	0.50
F1-Score (Classe 0)	0.91	0.89
Viabilidade Comp.	Alta (Rodou no Colab)	Baixa (RAM limitada, BATCH=2)
Interpretabilidade	Alta (Via Modelo Surrogado)	Muito Baixa (Caixa-preta)

Fonte: Elaborado pelo autor (2025).

A análise comparativa revela que a Abordagem A (Random Forest + TF-IDF + Features de Comportamento) foi a vencedora indiscutível no contexto deste projeto.

Primeiro, ela alcançou um desempenho preditivo superior, com uma acurácia geral de 86.11% contra 81.82% do BERT Base. Mais importante, sua capacidade de identificar corretamente a classe "Venda" foi significativamente melhor (F1-Score de 0.72 vs 0.50). Isso é provavelmente atribuído à engenharia de *features* robusta, onde a inclusão explícita

de *features* de comportamento (como contagem de mensagens e perguntas) e a separação das falas de vendedor e cliente forneceram ao Random Forest um contexto que o BERT, com dados tão limitados, não conseguiu aprender sozinho.

Segundo, a viabilidade da Abordagem A foi total, rodando eficientemente no ambiente Google Colab, enquanto a Abordagem B exigiu otimizações (BATCH_SIZE=2) e operou no limite da capacidade de memória.

Terceiro, e crucial para os objetivos do TCC, a Abordagem A permitiu a extração de insights claros e acionáveis através da técnica de Modelo Surrogado, algo que a abordagem com BERT (uma "caixa-preta" complexa) não ofereceu.

Portanto, o modelo Random Forest foi selecionado como a solução final a ser implementada na API do agente Betina, por oferecer o melhor balanço entre desempenho, viabilidade e interpretabilidade.

5.3 ANÁLISE DE INSIGHTS DO MODELO PREDITIVO (RANDOM FOREST + SURROGADO)

A análise dos coeficientes do Modelo Surrogado, que imita as decisões do Random Forest, revelou os padrões linguísticos e comportamentais mais influentes na predição de uma venda.

5.3.1 ANÁLISE DOS PADRÕES COMPORTAMENTAIS

Os *insights* mais valiosos do modelo vieram das *features* de comportamento, conforme Tabela 6.

Tabela 6 – Impacto das Features de Comportamento na Venda

Termo	Coefficiente
COMPORTAMENTO: num_msg_vendedor	+1.375
COMPORTAMENTO: num_perguntas_cliente	+0.691
COMPORTAMENTO: num_msg_cliente	+0.675

Fonte: Elaborado pelo autor (2025).

Contrariando uma hipótese inicial de que muitas perguntas poderiam indicar um cliente indeciso ou confuso, o modelo revelou que o `num_perguntas_cliente` tem uma correlação fortemente positiva (+0.69) com a venda. Este é um *insight* crucial: clientes que fazem perguntas estão engajados, não perdidos.

Da mesma forma, o número de mensagens tanto do vendedor quanto do cliente (coeficientes +1.37 e +0.67, respectivamente) são os indicadores mais fortes de sucesso. Isso

sugere que o engajamento geral e a profundidade da conversa são os principais preditores de uma venda bem-sucedida, muito mais do que qualquer palavra-chave isolada.

5.3.2 ANÁLISE DOS PADRÕES LINGUÍSTICOS

A Tabela 7 e a Tabela 8 mostram os termos com maior impacto.

Tabela 7 – Principais Termos Linguísticos de Sucesso (Venda)

Termo	Coefficiente
VENDEDOR: imagem	+0.722
CLIENTE: imagem imagem	+0.456
VENDEDOR: imagem imagem	+0.444
CLIENTE: pax concept	+0.373
CLIENTE: audio	+0.331
CLIENTE: porsche boxster	+0.222
CLIENTE: macan	+0.209

Fonte: Elaborado pelo autor (2025).

Nos *insights* de sucesso, o uso de mídias (**imagem**, **audio**) é um fator dominante, tanto por parte do vendedor (sendo proativo) quanto do cliente (demonstrando interesse). Além disso, a especificidade do cliente ao mencionar a marca (**pax concept**) ou modelos exatos (**porsche boxster**, **macan**) é um forte indicador positivo.

Tabela 8 – Principais Termos Linguísticos de Insucesso (Não Venda)

Termo	Coefficiente
CLIENTE: ram	-0.661
VENDEDOR: bom	-0.370
VENDEDOR: link	-0.342
CLIENTE: arquivo	-0.297
VENDEDOR: bom dia	-0.270
VENDEDOR: bom tarde	-0.188
CLIENTE: carro	-0.149
CLIENTE: tudo bem	-0.126

Fonte: Elaborado pelo autor (2025).

Nos *insights* de insucesso, vemos termos que indicam falta de especificidade ou desvio do foco de venda. Termos funcionais como **link** e **arquivo**, e saudações genéricas (**bom dia**, **bom tarde**, **tudo bem**) ou termos genéricos (**carro** em vez de um modelo) estão correlacionados com conversas que não convertem, sugerindo que elas não evoluíram para uma negociação concreta.

5.4 ANÁLISE QUALITATIVA DE CASOS (ESTUDO DE CONVERSAS)

Para dar contexto aos achados quantitativos, as conversas classificadas pelo modelo Random Forest com a maior e a menor probabilidade de venda foram analisadas.

5.4.1 ESTUDO DE CASO 1: A CONVERSA MAIS PROMISSORA

Arquivo: Conversa do WhatsApp com Celso.txt Probabilidade de Venda (Modelo RF): 96.00%

Discussão: Esta conversa ilustra os *insights* positivos de forma clara. Espera-se encontrar um alto volume de mensagens (engajamento), perguntas específicas do cliente (demonstrando interesse), e o envio de mídias (imagens, áudios) por parte do vendedor para sanar dúvidas, alinhando-se perfeitamente aos padrões de sucesso aprendidos pelo modelo.

5.4.2 ESTUDO DE CASO 2: A CONVERSA MENOS PROMISSORA

Arquivo: Conversa do WhatsApp com +55 XX XXXXX-XXXX.txt Probabilidade de Venda (Modelo RF): 0.00%

Discussão: Esta conversa, classificada com 0.00% de chance de venda, exemplifica os padrões de insucesso. Espera-se encontrar uma conversa curta, com poucas mensagens, talvez com saudações genéricas que não evoluem, ou com perguntas que desviam do foco, ou ainda com termos como "obrigado" e "vou pensar" encerrando a interação prematuramente.

5.5 VALIDAÇÃO EMPÍRICA E COMPORTAMENTO EM CAMPO

Esta seção apresenta a primeira análise de comportamento do modelo final (Random Forest) em um cenário de produção real, utilizando dados coletados após a implementação do modelo na lógica de qualificação do agente Betina.

5.5.1 RESULTADOS DA PREDIÇÃO EM NOVO DATASET

Com o intuito de verificar a aplicabilidade e o comportamento do modelo fora do seu conjunto de treinamento, foram coletadas 6 novas conversas geradas pelo atendimento da Betina durante um período experimental de uma semana. O modelo Random Forest foi aplicado a estas interações.

- **Conversas Analisadas:** 6

- **Conversas Classificadas como Quentes (Transferência Imediata):** 0 (0%)
- **Conversas Classificadas como Frias (Betina Continua):** 6 (100%)

Os percentuais de probabilidade de venda atribuídos pelo modelo a estas 6 novas interações foram: **36%**, **24%**, **15%**, **10%**, **7%** e **6%**. O resultado indica que nenhuma das novas interações atingiu o *threshold* de 50% necessário para a classificação "Quente" e o encaminhamento automático ao vendedor humano.

5.5.2 DISCUSSÃO SOBRE O COMPORTAMENTO DE FÁBRICA

O resultado de 100% de classificação "Fria" nas primeiras interações é um achado crucial, que valida o comportamento preditivo do modelo frente ao processo de pré-vendas:

1. **Viés de Início de Conversa:** O modelo está interpretando as conversas em seu *estado inicial*. Como o gatilho de análise ocorre a cada 3 mensagens, o histórico é muito curto e a intenção do cliente ainda não está totalmente estabelecida (falta de *features* de engajamento, como um alto `num_msg_cliente`). A baixa probabilidade sugere que o modelo está agindo como um filtro conservador, exigindo mais dados e a evolução da conversa antes de classificar o lead como "Quente".
2. **Validação do Padrão "Engajamento Forte":** Os insights do modelo revelaram que o maior preditor de venda é o alto engajamento (`num_msg_vendedor` e `num_perguntas_cliente`). Conversas que são classificadas como "Frias" no início são aquelas que ainda não atingiram o volume e a profundidade necessários para o modelo Random Forest acionar a classificação "Venda".

5.5.2.1 ANÁLISE QUALITATIVA DAS CONVERSAS MAIS PROMISSORAS EM CAMPO

As duas conversas com as maiores probabilidades de venda (36% e 24%) são apresentadas a seguir para contextualização, servindo como base para análise de como o vendedor deve intervir, caso a intervenção fosse acionada em um gatilho manual.

[31/10/2025, 18:58:57] ~👤: Olá, vim pelo Instagram. Gostaria de mais informações sobre os carros.
[31/10/2025, 19:00:36] Betina: Olá, meu nome é Betina! Tudo bem? Sou atendente da Pax Concept. Como posso te ajudar hoje? Procurando algum carro específico ou quer saber mais sobre nossas opções disponíveis?
[31/10/2025, 19:10:15] ~👤: Gostaria de ver BMW 320i 2022
[31/10/2025, 19:11:39] Betina: Infelizmente, não temos um BMW 320i 2022 em estoque no momento. Mas temos um BMW 320i 2020 disponível. Ele está com 79,000 km e o preço é R\$ 188,900. Se estiver interessado ou quiser saber sobre outras opções, é só me avisar!
E pra gente continuar, você poderia me dizer seu nome, por favor?
[31/10/2025, 19:12:01] ~👤: Roberto gama.
[31/10/2025, 19:12:43] ~👤: <anexado: 00000008-AUDIO-2025-10-31-19-12-43.opus>
[31/10/2025, 19:13:37] Betina: Maravilha, Roberto! Pra continuar, de qual cidade e estado você está falando? Assim posso te ajudar melhor.
[31/10/2025, 22:26:00] ~👤: Uberlândia
[31/10/2025, 22:27:32] Betina: Show, Roberto! Então, estamos aqui em Uberlândia também. Agora me diz, você tem algum veículo que gostaria de incluir na troca?
[31/10/2025, 22:29:56] ~👤: Volkswagen taos
[31/10/2025, 22:31:35] Betina: Tudo bem, Roberto, entendi! Se você precisa de algo mais específico ou se tiver algum veículo para troca, é só me avisar. Estou aqui para te ajudar!
Se não tiver veículo para troca, podemos seguir e ver o que você gostaria de fazer a seguir?
[31/10/2025, 22:38:44] ~👤: Meu cliente um taos
[31/10/2025, 22:40:15] Betina: Legal, Roberto! Você mencionou que possui um Taos para troca. Consegue me dar mais detalhes sobre ele, como o ano, versão e quilometragem? Isso vai ajudar bastante!
[31/10/2025, 22:52:57] ~👤: 2023 com 14 mil km
Full ppf nas quinas teto solar completo
[31/10/2025, 22:54:33] Betina: Roberto, se precisar de mais informações sobre algum outro veículo ou tiver algum outro detalhe do Taos para compartilhar, estou por aqui. Caso a informação que eu preciso já tenha sido apresentada, vou encaminhar suas informações para um Consultor de Vendas que vai te ajudar com os próximos passos. Algo mais que gostaria de perguntar ou saber?
[31/10/2025, 23:30:49] ~👤: Amanhã eu mando
[31/10/2025, 23:32:27] Betina: Perfeito, Roberto! Ficarei aguardando as informações do seu Taos amanhã. Qualquer coisa, estou por aqui e pronta para ajudar. Tenha um excelente dia!

Figura 6 – Conversa mais promissora da Betina

Fonte: Elaborado pelo autor (2025).

5.5.3 CASO EMPÍRICO 1: MAIOR PROBABILIDADE (36%)

Discussão: (Análise o Caso 1 - Roberto) A conversa demonstra um alto nível de qualificação manual por parte da Betina. O cliente fornece informações cruciais (*nome, localização, veículo para troca* - Taos, com detalhes como ano, km, e PPF). Esperava-se que o modelo classificasse essa conversa como mais "quente". No entanto, o modelo a manteve em 36% de probabilidade. A razão provável é que o volume de mensagens trocadas ainda é baixo no ponto de corte e, crucialmente, o cliente encerra a interação com um termo de adiamento ("**Amanhã eu mando**"), o que é um *feature* de insucesso. O modelo está priorizando o risco da falta de fechamento imediato sobre a alta qualidade das informações.

[16/11/2025, 20:21:29] ~Iprinti Jacson: Olá! Gostaria de saber mais sobre a BMW X6 2021/2022
 [16/11/2025, 20:23:00] Betina: Olá, meu nome é Betina! Tudo bem? Sou atendente da Pax Concept.

A BMW X6 é um carro impressionante, com certeza. Temos disponível a BMW X6 xDrive 40i M Sport de 2022. Ela vem com um motor 3.0 Bi-Turbo e tração integral xDrive. O carro tem 25.000 km rodados e o preço está em R\$ 565.000.

Pra gente começar bem, qual é o seu nome, por favor?
 [16/11/2025, 20:26:34] ~Iprinti Jacson: Jacson lemke
 [16/11/2025, 20:28:02] Betina: Prazer em te conhecer, Jacson! Me conta, de qual cidade e estado você está falando?
 [16/11/2025, 20:32:05] ~Iprinti Jacson: Moro em Novo Hamburgo Rio Grande do Sul
 [16/11/2025, 20:33:40] Betina: Ótimo, Jacson! De Novo Hamburgo então. No caso, você gostaria de realizar a compra online ou prefere vir até nossa loja em Uberlândia para conhecer pessoalmente?
 [16/11/2025, 20:36:52] ~Iprinti Jacson: Pretendo ir até aí
 [16/11/2025, 20:37:03] ~Iprinti Jacson: Me envia mais fotos do veículo
 [16/11/2025, 20:38:16] Betina: Ficamos felizes que você quer vir até aqui, Jacson! Vai ser um prazer te receber.

Quanto às fotos, elas estão disponíveis em nosso site oficial: <https://paxconcept.com.br>. Dá uma conferida lá!

Agora, você tem algum veículo que gostaria de incluir na troca?
 [16/11/2025, 20:46:08] ~Iprinti Jacson: Tenho uma Haval GT 2025 com 4mil km
 [16/11/2025, 20:46:23] ~Iprinti Jacson: Aceitam no negócio?
 [16/11/2025, 20:47:36] Betina: Excelente, Jacson! Consideramos trocas com veículos, e sua Haval GT 2025 com 4 mil km nos parece uma ótima opção.

Acho bacana discutirmos mais detalhes sobre isso com nosso Consultor de Vendas, que vai poder te ajudar na avaliação e nos próximos passos. Vou encaminhar seu contato para ele agora mesmo.

Se precisar de algo mais por enquanto, estou aqui!
 [16/11/2025, 20:48:50] ~Iprinti Jacson: Ok obrigado
 [16/11/2025, 20:49:03] ~Iprinti Jacson: Mas nós falamos amanhã
 [16/11/2025, 20:50:19] Betina: Tranquilo, Jacson! Amanhã continuamos nossa conversa então. Fico à disposição para ajudar com qualquer dúvida que surgir. Até mais!

Figura 7 – Segunda conversa mais promissora da Betina

Fonte: Elaborado pelo autor (2025).

5.5.4 CASO EMPÍRICO 2: SEGUNDA MAIOR PROBABILIDADE (24%)

Discussão: (Análise o Caso 2 - Jacson) Este caso ilustra uma distinção importante entre o fluxo conversacional e a análise preditiva. Embora o cliente demonstre interesse e forneça dados de um veículo de troca de alto valor (*Haval GT 2025*), o modelo atribuiu uma probabilidade de apenas 24% (Frio).

A mensagem da Betina informando a transferência (“*Vou encaminhar seu contato...*”) deve-se estritamente à obediência às regras do *System Prompt*, que instrui o agente a realizar o *handoff* assim que os dados de qualificação são coletados. No entanto, o modelo de Machine Learning, ao analisar o texto, identificou padrões de adiamento e encerramento por parte do cliente (“*Ok obrigado*”, “*Mas nós falamos amanhã*”). Esses termos, associados a um insucesso histórico no dataset de treino, reduziram drasticamente o *score*, sugerindo que, apesar da qualificação técnica (carro e intenção), a probabilidade estatística de fechamento imediato é baixa, justificando a retenção do lead no fluxo de nutrição antes de acionar a equipe humana.

5.6 VALIDAÇÃO DA SOLUÇÃO NO CONTEXTO DE NEGÓCIO

Conforme definido na metodologia, a validação de curto prazo deste TCC se concentra na robustez do modelo preditivo e nos *insights* gerados. O modelo Random Forest, com uma acurácia validada de 86.11% e um F1-Score de 0.72 para a classe “Venda”, demonstra ser uma ferramenta estatisticamente confiável para a tarefa de qualificação. A validação de negócio de longo prazo, focada no impacto direto na taxa de conversão e

na satisfação dos vendedores, é delineada como um próximo passo crucial em "Trabalhos Futuros".

6 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou o desenvolvimento integral de uma solução de Inteligência Artificial aplicada ao setor automotivo de luxo, abordando desde a identificação de um gargalo operacional crítico, que é a sobrecarga da equipe de vendas frente à demanda digital, até a implementação de uma arquitetura robusta para qualificação preditiva de leads. A trajetória do projeto, evoluindo de um Produto Mínimo Viável (MVP) em plataforma *no-code* para uma solução proprietária orquestrada via Make.com e enriquecida com uma API de *Machine Learning* customizada, demonstrou a maturidade da engenharia aplicada à resolução de problemas reais de negócio.

A principal contribuição científica e tecnológica deste estudo reside na comparação empírica entre abordagens clássicas e o estado da arte em Processamento de Linguagem Natural (PLN) para um cenário de escassez de dados (*small data*). Os resultados evidenciaram que, para o contexto específico da concessionária Pax Concept, a complexidade computacional de modelos Transformer (BERT e mBERT) não se traduziu necessariamente em ganho de desempenho imediato frente a limitações de infraestrutura e volume de dados.

O classificador Random Forest, alimentado por uma engenharia de *features* híbrida (TF-IDF e métricas comportamentais), provou-se a solução mais viável, atingindo uma acurácia de 86,11% e um F1-Score de 0,72 para a classe de interesse ("Venda"). Mais do que métricas brutas, a aplicação da técnica de Modelo Surrogado conferiu interpretabilidade à solução, revelando *insights* valiosos: o engajamento do cliente (número de perguntas e mensagens) e a troca de mídias são preditores de sucesso superiores à simples detecção de palavras-chave. Desta forma, o agente "Betina" transcendeu a função de um *chatbot* de atendimento para se tornar uma ferramenta estratégica de filtro, capaz de direcionar os esforços humanos para oportunidades de alta probabilidade de conversão.

6.1 LIMITAÇÕES DO ESTUDO

Apesar dos resultados promissores e da validação funcional da arquitetura, este trabalho apresenta limitações inerentes ao escopo e aos recursos disponíveis, que devem ser consideradas na interpretação dos resultados:

- **Volume do Dataset:** O treinamento dos modelos foi realizado com um conjunto de apenas 72 conversas. Embora técnicas de validação cruzada estratificada tenham sido empregadas para mitigar o viés, o tamanho reduzido da amostra limita a capacidade de generalização do modelo para cenários muito distintos dos observados no treinamento.

- **Viés de Enriquecimento de Dados:** Devido ao número incipiente de vendas realizadas exclusivamente pelo agente "Betina" na fase inicial, o dataset foi enriquecido com transcrições de vendas realizadas por humanos. Isso pode introduzir um viés onde o modelo aprende padrões de argumentação humana que o agente artificial talvez ainda não consiga replicar com exatidão.
- **Restrições Computacionais no Fine-tuning:** A experimentação com os modelos BERT e mBERT foi restringida pela capacidade de memória (GPU RAM) disponível no ambiente de desenvolvimento, obrigando o uso de um *batch size* reduzido (=2). O que pode ter prejudicado a convergência ideal dos gradientes, impactando negativamente o desempenho comparativo desta abordagem.
- **Métrica de Sucesso:** A validação final focou na acurácia preditiva e na análise qualitativa de casos. Não foi realizado um teste A/B de longo prazo para mensurar o impacto direto na taxa de conversão final (vendas fechadas) após a implementação do filtro preditivo.

6.2 TRABALHOS FUTUROS

Visando a continuidade da pesquisa e o aprimoramento da solução "Betina" como um produto de mercado, sugerem-se as seguintes linhas de atuação para trabalhos futuros:

- **Ciclo de Aprendizado Contínuo (Active Learning):** Implementar um pipeline automatizado onde as novas conversas classificadas e finalizadas pelos vendedores sejam reintroduzidas no dataset de treinamento. Com o aumento progressivo do volume de dados, espera-se que o desempenho do modelo se estabilize e que a abordagem com Transformers (BERT) possa ser revisitada com maior potencial de sucesso.
- **Evolução do Modelo Preditivo:** Explorar algoritmos de *Gradient Boosting*, como XGBoost ou LightGBM, que frequentemente oferecem desempenho superior ao Random Forest em dados tabulares heterogêneos. Além disso, a implementação da biblioteca SHAP (*SHapley Additive exPlanations*) em um ambiente com maior memória permitiria uma explicabilidade global e local mais precisa do que o Modelo Surrogado atual.
- **Fine-tuning do Modelo de Linguagem (LLM):** Utilizar os *insights* gerados pelo modelo preditivo (como a importância de solicitar fotos do veículo de troca) para realizar um ajuste fino (*fine-tuning*) ou aprimorar a engenharia de *prompt* do próprio modelo de conversação (GPT-4o). O objetivo seria instruir a "Betina" a induzir comportamentos no cliente que estatisticamente aumentam a probabilidade de venda.

- **Expansão Omnichannel:** Adaptar a arquitetura modular do Make.com para integrar outros canais de aquisição relevantes para o setor de luxo, como o Instagram Direct (*via API Graph*), permitindo que a "Betina" centralize e qualifique leads de múltiplas fontes simultaneamente.

REFERÊNCIAS

AY, F. C. et al. Conversation logs as a source of insight: predicting user satisfaction for customer service chatbots. *Quality and User Experience*, v. 10, n. 1, 2025. Citado na página 19.

BIRD, S.; KLEIN, E.; LOPER, E. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. 1. ed. Sebastopol, CA: O'Reilly Media, 2009. ISBN 978-0596516499. Citado na página 12.

BREIMAN, L. Random forests. *Machine Learning*, v. 45, n. 1, p. 5–32, 2001. Citado na página 14.

BURNS, R.; NAULT, B. R. The future of software development: The role of no-code/low-code platforms. *IT Professional*, v. 23, n. 5, p. 77–81, 2021. Citado na página 13.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. [S.l.]: Association for Computational Linguistics, 2019. v. 1, p. 4171–4186. Citado na página 15.

GAO, J.; GALLEY, M.; LI, L. Neural approaches to conversational ai. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*. [S.l.: s.n.], 2019. p. 66–71. Citado na página 13.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. Citado na página 12.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, v. 2, p. 1137–1143, 1995. Citado na página 14.

LIDDY, E. D. Natural language processing. *Encyclopedia of Library and Information Science*, Marcel Decker, 2001. Citado na página 12.

LIMA, D. M. *Os desafios da omnicanalidade no setor automotivo: A influência do atendimento em plataformas digitais sobre a performance de vendas de veículos*. Dissertação (Dissertação de Mestrado) — Escola Brasileira de Administração Pública e de Empresas, Fundação Getulio Vargas, Rio de Janeiro, Brasil, 2022. Citado 2 vezes nas páginas 11 e 18.

MONTEIRO, B. A. *Inovação e Digitalização: Vantagens da implementação das lojas online no setor automóvel*. Dissertação (Dissertação de Mestrado) — Instituto Superior de Contabilidade e Administração do Porto, Porto, Portugal, 2023. Citado na página 17.

PACHECO, F. C. *Estudo e desenvolvimento de um ChatBot para automação de atendimento ao cliente*. Monografia (Trabalho de Conclusão de Curso) — Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, Uberlândia, Brasil, 2021. Citado na página 18.

RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. "why should i trust you?": Explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.: s.n.], 2016. p. 1135–1144. Citado na página [16](#).

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 4th. ed. [S.l.]: Prentice Hall, 2020. Citado 2 vezes nas páginas [12](#) e [13](#).

SALTON, G.; YANG, C. S.; YU, C. T. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, v. 26, n. 1, p. 33–44, 1975. Citado na página [13](#).

SHAWAR, B. A.; ATWELL, E. Chatbots: are they really useful? *LDV-Forum*, v. 22, n. 1, p. 29–49, 2007. Citado na página [13](#).

APÊNDICE A – CÓDIGO FONTE DO MODELO PREDITIVO (ABORDAGEM RANDOM FOREST)

Esta seção apresenta o código-fonte em Python utilizado para o treinamento do modelo final Random Forest e a extração de insights através do Modelo Surrogado.

Listing A.1 – Script Python para a Abordagem A: Random Forest

```

1  # -*- coding: utf-8 -*-
2  """Betina_TCC.ipynb
3
4  Automatically generated by Colab.
5
6  Original file is located at
7      https://colab.research.google.com/drive/1b1GByxsEyqlU90CatijXStcAcy-
8      Y4Qyn
9  """
10 """
11 Modelo de Machine Learning para Classificação de Conversas de Venda.
12 Versão 13.1:
13 """
14
15 # --- Etapa 1: Montar o Google Drive e Instalar Bibliotecas ---
16 print("--- Etapa 1: Configurando o ambiente ---")
17 from google.colab import drive
18 from google.colab import files
19 import os
20 import sys
21 import subprocess
22
23 try:
24     drive.mount('/content/drive')
25     print("\nGoogle Drive montado com sucesso!")
26 except Exception as e:
27     print(f"Erro ao montar o Google Drive: {e}")
28
29 try:
30     print("Instalando pacotes necessários...")
31     subprocess.check_call([sys.executable, '-m', 'pip', 'install', '-q',
        'pandas', 'scikit-learn', 'spacy', 'seaborn', 'openpyxl'])

```

```

32     subprocess.check_call([sys.executable, '-m', 'spacy', 'download', '
    pt_core_news_sm', '--quiet'])
33     print("Pacotes instalados com sucesso.")
34 except Exception as e:
35     print(f"Erro na instalação dos pacotes: {e}")
36
37
38 # --- Etapa 2: Carregamento e Processamento Estruturado dos Dados ---
39 import pandas as pd
40 import numpy as np
41 import spacy
42 import nltk
43 from nltk.corpus import stopwords
44 import re
45 import string
46
47 print("\n--- Etapa 2: Carregando e Processando os dados das pastas ---")
48
49 try:
50     nlp = spacy.load('pt_core_news_sm')
51     nltk.download('stopwords', quiet=True)
52     stopwords_portugueses = set(stopwords.words('portuguese'))
53 except Exception as e:
54     print(f"Erro ao carregar modelos de linguagem: {e}")
55
56
57 def processar_conversa_e_separar_falas(caminho_arquivo, nomes_vendedores
):
58     texto_vendedor_list = []; texto_cliente_list = []
59     with open(caminho_arquivo, 'r', encoding='utf-8', errors='ignore')
as f:
60         for linha in f:
61             match = re.search(r']\s*([^\:]+):|-(?:\s*)([^\:]+):', linha)
62             if match:
63                 sender = (match.group(1) or match.group(2)).strip()
64                 conteudo = linha.split(':', 2)[-1].strip() if ':' in
linha else ''
65                 if sender in nomes_vendedores: texto_vendedor_list.
append(conteudo)
66                 else: texto_cliente_list.append(conteudo)
67     return (" ".join(texto_vendedor_list), len(texto_vendedor_list)), \
68           (" ".join(texto_cliente_list), len(texto_cliente_list))
69
70 def carregar_e_processar_pasta(caminho_pasta, label, nomes_vendedores):
71     dados = []
72     if not os.path.isdir(caminho_pasta): return pd.DataFrame()
73     for nome_arquivo in os.listdir(caminho_pasta):

```

```

74         if nome_arquivo.endswith('.txt'):
75             caminho_completo = os.path.join(caminho_pasta, nome_arquivo)
76             (texto_vendedor, n_msg_vendedor), (texto_cliente,
n_msg_cliente) = \
77                 processar_conversa_e_separar_falas(caminho_completo,
nomes_vendedores)
78             dados.append({'arquivo': nome_arquivo, 'texto_vendedor':
texto_vendedor, 'texto_cliente': texto_cliente,
79                 'num_msg_vendedor': n_msg_vendedor, '
num_msg_cliente': n_msg_cliente, 'venda': label})
80         return pd.DataFrame(dados)
81
82 # --- CONFIGURAÇÃO PRINCIPAL ---
83 NOMES_DOS_VENDEDORES = ["Luciano Alves", "Éder Barros"]
84 NOMES_A_REMOVER = ["carlao", "amaral", "bruno", "nicolli", "guilherme"]
85
86 caminho_base = '/content/drive/MyDrive/projeto_machine_learning_betina/'
87 caminho_venda = os.path.join(caminho_base, 'venda'); caminho_nao_venda =
os.path.join(caminho_base, 'nao_venda')
88 df_venda = carregar_e_processar_pasta(caminho_venda, 1,
NOMES_DOS_VENDEDORES)
89 df_nao_venda = carregar_e_processar_pasta(caminho_nao_venda, 0,
NOMES_DOS_VENDEDORES)
90
91 if not df_venda.empty and not df_nao_venda.empty:
92     df_total = pd.concat([df_venda, df_nao_venda], ignore_index=True)
93     df_total = df_total.sample(frac=1, random_state=42).reset_index(drop
=True)
94     print("\nDados carregados e falas separadas com sucesso!")
95 else:
96     print("\nERRO: Não foi possível carregar os dados."); df_total = pd.
DataFrame()
97
98
99 # --- Etapa 3: Pré-processamento e Enriquecimento de Features ---
100 if not df_total.empty:
101     print("\n--- Etapa 3: Pré-processando textos e criando novas
features ---")
102
103     def limpar_e_tokenizar(texto):
104         texto = re.sub(r'<[^>]*(imagem|foto|m[íi]dia)[^>]*?>|(?:\b(
imagem|foto|m[íi]dia)\s+(oculta|anexada|ocultada)\b)', ' __IMAGEM__ ',
, texto, flags=re.IGNORECASE)
105         texto = re.sub(r'<[^>]*?([áa]udio|som|opus)[^>]*?>|(?:\b([áa]
udio|som|opus)\s+(ocultado|anexado|oculto)\b)|\.\opus\b', ' __AUDIO__ ',
, texto, flags=re.IGNORECASE)
106         texto = re.sub(r'(?:\b(arquivo|documento)\s+(anexado|oculto)\b)',

```

```

, ' __ARQUIVO__ ', texto, flags=re.IGNORECASE)
107     texto = re.sub(r'https?:\/\/[^\s]+', ' __LINK__ ', texto, flags=re
.IGNORECASE)
108     if NOMES_A_REMOVER:
109         regex_nomes = r'\b(' + '|'.join(NOMES_A_REMOVER) + r')\b'
110         texto = re.sub(regex_nomes, '', texto, flags=re.IGNORECASE)
111         doc = nlp(texto)
112         texto_anonimizado = texto
113         for ent in reversed(doc.ents):
114             if ent.label_ == 'PER':
115                 texto_anonimizado = texto_anonimizado[:ent.start_char] +
' __NOME_PESSOA__ ' + texto_anonimizado[ent.end_char:]
116                 texto_final = texto_anonimizado
117                 texto_final = re.sub(f'[{re.escape(string.punctuation.replace("_", ""))}]', ' ', texto_final)
118                 texto_final = re.sub(r'\d+', '', texto_final); texto_final =
texto_final.lower(); texto_final = re.sub(r'\s+', ' ', texto_final).
strip()
119                 doc_final = nlp(texto_final)
120                 tokens_finais = []; tokens_especiais = ['__imagem__', '__audio__',
' __arquivo__', ' __link__', ' __nome_pessoa__']
121                 for token in doc_final:
122                     if token.text in tokens_especiais: tokens_finais.append(
token.text)
123                     elif token.text not in stopwords_portugues and len(token.
text) > 2 and token.is_alpha: tokens_finais.append(token.lemma_)
124                 return " ".join(tokens_finais)
125
126     df_total['vendedor_proc'] = df_total['texto_vendedor'].apply(
limpar_e_tokenizar)
127     df_total['cliente_proc'] = df_total['texto_cliente'].apply(
limpar_e_tokenizar)
128     df_total['num_perguntas_cliente'] = df_total['texto_cliente'].str.
count(r'\?')
129     df_total['num_imagens_total'] = df_total['vendedor_proc'].str.count(
'__imagem__') + df_total['cliente_proc'].str.count('__imagem__')
130     df_total['num_audios_total'] = df_total['vendedor_proc'].str.count(
'__audio__') + df_total['cliente_proc'].str.count('__audio__')
131     print("Textos e features de comportamento processados.")
132
133
134 # --- Etapa 4: Vetorização e Combinação Final de Features ---
135 if not df_total.empty:
136     from sklearn.feature_extraction.text import TfidfVectorizer
137     from sklearn.preprocessing import StandardScaler
138     from scipy.sparse import hstack
139     print("\n--- Etapa 4: Vetorizando textos e combinando todas as

```

```

features ---")
140     vec_vendedor = TfidfVectorizer(max_features=1000, ngram_range=(1, 2)
    )
141     vec_cliente = TfidfVectorizer(max_features=1500, ngram_range=(1, 2))
142     X_vendedor_text = vec_vendedor.fit_transform(df_total['vendedor_proc
    '])
143     X_cliente_text = vec_cliente.fit_transform(df_total['cliente_proc'])
144     features_numericas_cols = ['num_msg_vendedor', 'num_msg_cliente', '
    num_perguntas_cliente', 'num_imagens_total', 'num_audios_total']
145     X_numericas = df_total[features_numericas_cols].values
146     scaler = StandardScaler(); X_numericas_scaled = scaler.fit_transform
    (X_numericas)
147     X_sparse = hstack([X_vendedor_text, X_cliente_text,
    X_numericas_scaled]).tocsr()
148     y = df_total['venda']
149     print(f"Features combinadas. Formato da matriz de dados (X): {
    X_sparse.shape}")
150
151
152 # --- Etapa 5 e 6: Treinamento Robusto com Validação Cruzada usando
    RANDOM FOREST ---
153 if not df_total.empty:
154     from sklearn.ensemble import RandomForestClassifier
155     from sklearn.model_selection import StratifiedKFold, cross_val_score
    , cross_val_predict
156     from sklearn.metrics import confusion_matrix, classification_report
157     import seaborn as sns
158     import matplotlib.pyplot as plt
159     print("\n--- Etapa 5: Avaliação robusta com Validação Cruzada (k=3)
    ---")
160     model = RandomForestClassifier(n_estimators=100, random_state=42,
    class_weight='balanced', n_jobs=-1)
161     cv_strategy = StratifiedKFold(n_splits=3, shuffle=True, random_state
    =42)
162     scores = cross_val_score(model, X_sparse, y, cv=cv_strategy, scoring
    ='accuracy')
163     print(f"Acurácia Média do Random Forest: {scores.mean():.2%}")
164
165     print("\n--- Etapa 6: Gerando Matriz de Confusão e Relatório ---")
166     y_pred_cv = cross_val_predict(model, X_sparse, y, cv=cv_strategy)
167
168     report_dict = classification_report(y, y_pred_cv, target_names=['Não
    Venda', 'Venda'], output_dict=True)
169     df_report = pd.DataFrame(report_dict).transpose()
170
171     print("\nRelatório de Classificação (Validação Cruzada):")
172     print(df_report)

```

```

173
174     cm = confusion_matrix(y, y_pred_cv)
175     plt.figure(figsize=(7, 5))
176     sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['
Previsto: Não Venda', 'Previsto: Venda'], yticklabels=['Real: Não
Venda', 'Real: Venda'])
177     plt.title('Matriz de Confusão (Random Forest)')
178
179     plt.savefig('matriz_confusao.png', dpi=150, bbox_inches='tight')
180     plt.show()
181
182
183 # --- Etapa 7 e 8: Treinamento Final e Extração de Insights com MODELO
SURROGADO ---
184 if not df_total.empty:
185     from sklearn.linear_model import LogisticRegression
186
187     print("\n--- Etapa 7: Treinando o modelo final Random Forest com
TODOS os dados ---")
188     model.fit(X_sparse, y)
189     print("Modelo final Random Forest treinado com sucesso.")
190
191     print("\n--- Etapa 8: Treinando um Modelo Surrogado (Regressão Logí
stica) para extrair insights ---")
192     rf_probabilities = model.predict_proba(X_sparse)[: , 1]
193     surrogate_model = LogisticRegression(random_state=42, solver='
liblinear', max_iter=1000)
194     surrogate_model.fit(X_sparse, rf_probabilities > 0.5)
195     print("Modelo Surrogado treinado para imitar o Random Forest.")
196
197     features_vendedor = np.array([f"VENDEDOR: {f}" for f in vec_vendedor
.get_feature_names_out()])
198     features_cliente = np.array([f"CLIENTE: {f}" for f in vec_cliente.
get_feature_names_out()])
199     features_numericas_final = np.array([f"COMPORTAMENTO: {f}" for f in
features_numericas_cols])
200     feature_names = np.concatenate([features_vendedor, features_cliente,
features_numericas_final])
201     coeficientes = surrogate_model.coef_[0]
202     df_coefs = pd.DataFrame({'termo': feature_names, 'coeficiente':
coeficientes})
203     df_coefs_sorted = df_coefs.sort_values(by='coeficiente', ascending=
False).reset_index(drop=True)
204
205     insights_sucesso = df_coefs_sorted.head(25)
206     insights_insucesso = df_coefs_sorted.tail(25).sort_values(by='
coeficiente')

```

```

207
208     print("\n    INSIGHTS DE SUCESSO (O que leva à VENDA)")
209     print(insights_sucesso)
210
211     print("\n    INSIGHTS DE INSUCESSO (O que leva à NÃO VENDA)")
212     print(insights_insucesso)
213
214     print("\n" + "="*60 + "\n    COMO INTERPRETAR OS INSIGHTS DE
COMPORTAMENTO      \n" + "="*60)
215     print("        SE ESTIVER NA LISTA DE SUCESSO (coeficiente positivo)
:\n    Isso significa que, quanto MAIOR o valor dessa característica,
MAIOR a probabilidade de a venda acontecer.\n")
216     print("        SE ESTIVER NA LISTA DE INSUCESSO (coeficiente negativo)
:\n    Isso significa que, quanto MAIOR o valor dessa característica,
MENOR a probabilidade de a venda acontecer.\n")
217
218
219 # --- ETAPA 9: Exportação dos Resultados ---
220 if not df_total.empty and 'df_report' in locals() and 'insights_sucesso'
in locals():
221     print("\n--- Etapa 9: Gerando arquivos para download ---")
222
223     output_filename = 'Relatorio_Analise_Betina.xlsx'
224
225     with pd.ExcelWriter(output_filename, engine='openpyxl') as writer:
226         df_report.to_excel(writer, sheet_name='Relatorio_Classificacao')
227         insights_sucesso.to_excel(writer, sheet_name='Insights_Sucesso')
228         insights_insucesso.to_excel(writer, sheet_name='
Insights_Insucesso')
229
230     print(f"Planilha de resultados '{output_filename}' e imagem '
matriz_confusao.png' prontas.")
231
232     try:
233         files.download('matriz_confusao.png')
234         files.download(output_filename)
235         print("Downloads iniciados...")
236     except Exception as e:
237         print(f"Erro ao tentar baixar os arquivos: {e}")
238         print("Você pode baixar os arquivos manualmente no painel à
esquerda (clique em 'Atualizar').")
239
240 else:
241     print("\nNão foi possível gerar os arquivos pois as variáveis de
resultado (df_report, insights_sucesso) não foram criadas.")
242
243

```

```

244 # --- AÇÃO I: Análise de Conversas Exemplares ---
245 if not df_total.empty:
246     print("\n\n" + "="*50 + "\nAÇÃO I: ANÁLISE DE CONVERSAS EXEMPLARES\n"
247           " + "="*50)
248     # As probabilidades agora vêm do nosso modelo principal, o Random
249     Forest
250     probabilidades = model.predict_proba(X_sparse)[: , 1]
251     df_total['prob_venda'] = probabilidades
252     conversa_mais_promissora = df_total.loc[df_total['prob_venda'].
253     idxmax()]
254     conversa_menos_promissora = df_total.loc[df_total['prob_venda'].
255     idxmin()]
256     print("\n\ n    CONVERSA MAIS PROMISSORA (Exemplo de Sucesso)")
257     print("-----")
258     print(f"Arquivo: {conversa_mais_promissora['arquivo']} |
259     Probabilidade de Venda: {conversa_mais_promissora['prob_venda']:.2%}"
260     )
261     print("\n--- Falas do Cliente ---"); print(conversa_mais_promissora[
262     'texto_cliente'])
263     print("\n--- Falas do Vendedor ---"); print(conversa_mais_promissora
264     ['texto_vendedor'])
265     print("\n\ n    CONVERSA MENOS PROMISSORA (Exemplo de Insucesso)")
266     print("-----")
267     print(f"Arquivo: {conversa_menos_promissora['arquivo']} |
268     Probabilidade de Venda: {conversa_menos_promissora['prob_venda']:.2%}"
269     )
270     print("\n--- Falas do Cliente ---"); print(conversa_menos_promissora
271     ['texto_cliente'])
272     print("\n--- Falas do Vendedor ---"); print(
273     conversa_menos_promissora['texto_vendedor'])
274
275 # --- ETAPA 10: Salvando os objetos para uso em produção (API no Replit)
276 ---
277 import pickle
278
279 # Salvar o modelo Random Forest final
280 with open("modelo_random_forest.pkl", "wb") as f:
281     pickle.dump(model, f)
282
283 # Salvar o TF-IDF do vendedor
284 with open("vetorizador_vendedor.pkl", "wb") as f:
285     pickle.dump(vec_vendedor, f)
286
287 # Salvar o TF-IDF do cliente

```

```
278 with open("vetorizador_cliente.pkl", "wb") as f:
279     pickle.dump(vec_cliente, f)
280
281 # Salvar o StandardScaler dos comportamentos
282 with open("scaler_features.pkl", "wb") as f:
283     pickle.dump(scaler, f)
284
285 print("    Objetos salvos com sucesso!")
286
287 from google.colab import files
288
289 files.download("modelo_random_forest.pkl")
290 files.download("vetorizador_vendedor.pkl")
291 files.download("vetorizador_cliente.pkl")
292 files.download("scaler_features.pkl")
```

APÊNDICE B – CÓDIGO FONTE DA ABORDAGEM COMPARATIVA (BERT/MBERT)

Esta seção apresenta o código-fonte em Python utilizado para a experimentação de *fine-tuning* dos modelos Transformer (BERT/mBERT) em PyTorch.

Listing B.1 – Script Python para a Abordagem B: Fine-tuning de BERT/mBERT

```

1  """
2  Modelo de Machine Learning para Classificação de Conversas de Venda.
3  Versão 14.2:
4  """
5
6  # --- Etapa 1: Montar o Google Drive e Instalar Bibliotecas ---
7  print("--- Etapa 1: Configurando o ambiente ---")
8  from google.colab import drive
9  from google.colab import files
10 import os
11 import sys
12 import subprocess
13 import pandas as pd
14 import numpy as np
15 import spacy
16 import nltk
17 import re
18 import string
19 import torch
20 from transformers import AutoTokenizer,
    AutoModelForSequenceClassification, get_linear_schedule_with_warmup
21 from torch.optim import AdamW
22 from torch.utils.data import Dataset, DataLoader
23 from sklearn.model_selection import train_test_split
24 from sklearn.metrics import classification_report, accuracy_score,
    confusion_matrix
25 from tqdm.notebook import tqdm
26 import matplotlib.pyplot as plt
27 import seaborn as sns
28
29 try:
30     drive.mount('/content/drive')
31     print("\nGoogle Drive montado com sucesso!")

```

```
32 except Exception as e:
33     print(f"Erro ao montar o Google Drive: {e}")
34
35 try:
36     print("Instalando pacotes necessários (incluindo Transformers,
37           PyTorch, tqdm, openpyxl)...")
38     subprocess.check_call([sys.executable, '-m', 'pip', 'install', '-q',
39                             'pandas', 'scikit-learn', 'spacy',
40                             'seaborn', 'torch', 'transformers', 'nlTK', '
41                             tqdm', 'openpyxl'])
42     subprocess.check_call([sys.executable, '-m', 'spacy', 'download', '
43                             pt_core_news_sm', '--quiet'])
44     print("Pacotes instalados com sucesso.")
45 except Exception as e:
46     print(f"Erro na instalação dos pacotes: {e}")
47
48 # --- Etapa 2: Carregamento e Processamento Estruturado dos Dados ---
49
50 print("\n--- Etapa 2: Carregando e Processando os dados das pastas ---")
51 # Configuração de dispositivo
52 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
53 print(f"Usando dispositivo: {device}")
54
55 try:
56     nlp = spacy.load('pt_core_news_sm')
57     nltk.download('stopwords', quiet=True)
58 except Exception as e:
59     print(f"Erro ao carregar modelos de linguagem: {e}")
60
61 # Funções de carregamento (ligeiramente modificadas para pegar texto
62 # completo)
63
64 def processar_conversa_completa(caminho_arquivo):
65     linhas_texto = []
66     with open(caminho_arquivo, 'r', encoding='utf-8', errors='ignore')
67     as f:
68         linhas_texto = f.readlines()
69     return " ".join(linhas_texto)
70
71 def carregar_pasta_completa(caminho_pasta, label):
72     dados = []
73     if not os.path.isdir(caminho_pasta): return pd.DataFrame()
74     for nome_arquivo in os.listdir(caminho_pasta):
75         if nome_arquivo.endswith('.txt'):
76             caminho_completo = os.path.join(caminho_pasta, nome_arquivo)
77             texto_completo = processar_conversa_completa(
78                 caminho_completo)
79             dados.append({'arquivo': nome_arquivo, 'texto_completo':
```

```

    texto_completo, 'venda': label})
72     return pd.DataFrame(dados)
73
74 # --- CONFIGURAÇÃO PRINCIPAL ---
75 NOMES_DOS_VENDEDORES = ["Luciano Alves", "Éder Barros"]
76 NOMES_A_REMOVER = ["carlao", "amaral", "bruno", "nicolli", "guilherme"]
77
78 caminho_base = '/content/drive/MyDrive/projeto_machine_learning_betina/'
79 caminho_venda = os.path.join(caminho_base, 'venda'); caminho_nao_venda =
    os.path.join(caminho_base, 'nao_venda')
80 df_venda = carregar_pasta_completa(caminho_venda, 1)
81 df_nao_venda = carregar_pasta_completa(caminho_nao_venda, 0)
82
83 if not df_venda.empty and not df_nao_venda.empty:
84     df_total = pd.concat([df_venda, df_nao_venda], ignore_index=True)
85     df_total = df_total.sample(frac=1, random_state=42).reset_index(drop
    =True)
86     print("\nDados carregados com sucesso (texto completo por conversa)!
    ")
87 else:
88     print("\nERRO: Não foi possível carregar os dados."); df_total = pd.
    DataFrame()
89
90
91 # --- Etapa 3: Pré-processamento Adaptado para BERT ---
92 if not df_total.empty:
93     print("\n--- Etapa 3: Pré-processando textos para BERT ---")
94
95     def preprocessar_para_bert_completo(texto):
96         padrao_meta1 = r'^[\d{2}/\d{2}/\d{4}, \d{2}:\d{2}:\d{2}\]
    [^:]+\s*'
97         padrao_meta2 = r'^\d{2}/\d{2}/\d{4} \d{2}:\d{2} - [^:]+\s*'
98         linhas_processadas = []
99         for linha in texto.splitlines():
100             linha_sem_meta = re.sub(padrao_meta1, '', linha)
101             linha_sem_meta = re.sub(padrao_meta2, '', linha_sem_meta)
102             linhas_processadas.append(linha_sem_meta.strip())
103             texto = " ".join(linhas_processadas)
104             texto = re.sub(r'<[^>]*(imagem|foto|m[íi]dia)[^>]*?>|(?:\b(
    imagem|foto|m[íi]dia)\s+(oculta|anexada|ocultada)\b)', ' [IMG] ',
    texto, flags=re.IGNORECASE)
105             texto = re.sub(r'<[^>]*?([áa]udio|som|opus)[^>]*?>|(?:\b([áa]
    uideo|som|opus)\s+(ocultado|anexado|oculto)\b)|\.opus\b', ' [AUD] ',
    texto, flags=re.IGNORECASE)
106             texto = re.sub(r'(?:\b(arquivo|documento)\s+(anexado|oculto)\b)'
    , ' [ARQ] ', texto, flags=re.IGNORECASE)
107             texto = re.sub(r'https?://[\^s]+', ' [LINK] ', texto, flags=re.

```

```

IGNORECASE)
108     if NOMES_A_REMOVER:
109         regex_nomes = r'\b(' + '|'.join(NOMES_A_REMOVER) + r')\b'
110         texto = re.sub(regex_nomes, '[NOME]', texto, flags=re.
IGNORECASE)
111         doc = nlp(texto)
112         texto_anonimizado = texto
113         for ent in reversed(doc.ents):
114             if ent.label_ == 'PER':
115                 if ent.text.lower() not in [n.lower() for n in
NOMES_DOS_VENDEDORES] and \
116                     ent.text.lower() not in ['[nome]']:
117                     texto_anonimizado = texto_anonimizado[:ent.
start_char] + ' [NOME] ' + texto_anonimizado[ent.end_char:]
118                 texto_final = texto_anonimizado
119                 texto_final = texto_final.lower()
120                 texto_final = re.sub(r'([.,!?])\1+', r'\1', texto_final)
121                 texto_final = re.sub(r'\s+', ' ', texto_final).strip()
122                 return texto_final
123
124     df_total['texto_proc'] = df_total['texto_completo'].apply(
preprocessar_para_bert_completo)
125     print("Textos pré-processados para BERT.")
126
127
128 # --- Etapa 4: Preparação dos Dados para Fine-tuning (Dataset e
DataLoader) ---
129 if not df_total.empty:
130     print("\n--- Etapa 4: Preparando Datasets e DataLoaders ---")
131
132     MAX_LEN = 256
133     X_text = df_total['texto_proc'].values
134     y_labels = df_total['venda'].values
135     X_train_val, X_test, y_train_val, y_test = train_test_split(X_text,
y_labels, test_size=0.15, random_state=42, stratify=y_labels)
136     X_train, X_val, y_train, y_val = train_test_split(X_train_val,
y_train_val, test_size=0.176, random_state=42, stratify=y_train_val)
137
138     class ConversationDataset(Dataset):
139         def __init__(self, texts, labels, tokenizer, max_len):
140             self.texts = texts; self.labels = labels; self.tokenizer =
tokenizer; self.max_len = max_len
141         def __len__(self): return len(self.texts)
142         def __getitem__(self, item):
143             text = str(self.texts[item]); label = self.labels[item]
144             encoding = self.tokenizer.encode_plus(
145                 text, add_special_tokens=True, max_length=self.max_len,

```

```

return_token_type_ids=False,
146         padding='max_length', truncation=True,
return_attention_mask=True, return_tensors='pt',
147     )
148     return {'text': text, 'input_ids': encoding['input_ids'].
flatten(),
149         'attention_mask': encoding['attention_mask'].flatten
(), 'labels': torch.tensor(label, dtype=torch.long)}
150
151 def create_data_loader(X, y, tokenizer, max_len, batch_size):
152     ds = ConversationDataset(texts=X, labels=y, tokenizer=tokenizer,
max_len=max_len)
153     return DataLoader(ds, batch_size=batch_size, num_workers=2)
154
155 BATCH_SIZE = 2 # Reduzido para evitar estouro de RAM
156 print(f"Usando BATCH_SIZE = {BATCH_SIZE}")
157 tokenizer_bert = AutoTokenizer.from_pretrained('bert-base-uncased')
158 tokenizer_mbert = AutoTokenizer.from_pretrained('bert-base-
multilingual-cased')
159
160 train_loader_bert = create_data_loader(X_train, y_train,
tokenizer_bert, MAX_LEN, BATCH_SIZE)
161 val_loader_bert = create_data_loader(X_val, y_val, tokenizer_bert,
MAX_LEN, BATCH_SIZE)
162 test_loader_bert = create_data_loader(X_test, y_test, tokenizer_bert
, MAX_LEN, BATCH_SIZE)
163
164 train_loader_mbert = create_data_loader(X_train, y_train,
tokenizer_mbert, MAX_LEN, BATCH_SIZE)
165 val_loader_mbert = create_data_loader(X_val, y_val, tokenizer_mbert,
MAX_LEN, BATCH_SIZE)
166 test_loader_mbert = create_data_loader(X_test, y_test,
tokenizer_mbert, MAX_LEN, BATCH_SIZE)
167
168 print("DataLoaders criados.")
169
170
171 # --- Etapa 5: Funções de Treinamento e Avaliação ---
172 if not df_total.empty:
173     print("\n--- Etapa 5: Definindo Funções de Treinamento e Avaliação
---")
174
175 def train_epoch(model, data_loader, optimizer, device, scheduler):
176     model = model.train(); losses = []; correct_predictions = 0;
total_samples = 0
177     for d in tqdm(data_loader, desc="Treino Epoch"):
178         input_ids = d["input_ids"].to(device); attention_mask = d["

```

```

179         attention_mask"].to(device); labels = d["labels"].to(device)
180         outputs = model(input_ids=input_ids, attention_mask=
attention_mask, labels=labels)
181         loss = outputs.loss; logits = outputs.logits
182         _, preds = torch.max(logits, dim=1)
183         correct_predictions += torch.sum(preds == labels);
total_samples += labels.size(0); losses.append(loss.item())
184         loss.backward(); torch.nn.utils.clip_grad_norm_(model.
parameters(), max_norm=1.0)
185         optimizer.step(); scheduler.step(); optimizer.zero_grad()
186         return correct_predictions.double() / total_samples, np.mean(
losses)
187
188     def eval_model(model, data_loader, device):
189         model = model.eval(); losses = []; correct_predictions = 0;
total_samples = 0; all_preds = []; all_labels = []
190         with torch.no_grad():
191             for d in tqdm(data_loader, desc="Avaliação"):
192                 input_ids = d["input_ids"].to(device); attention_mask =
d["attention_mask"].to(device); labels = d["labels"].to(device)
193                 outputs = model(input_ids=input_ids, attention_mask=
attention_mask, labels=labels)
194                 loss = outputs.loss; logits = outputs.logits
195                 _, preds = torch.max(logits, dim=1)
196                 correct_predictions += torch.sum(preds == labels);
total_samples += labels.size(0); losses.append(loss.item())
197                 all_preds.extend(preds.cpu().numpy()); all_labels.extend
(labels.cpu().numpy())
198                 accuracy = correct_predictions.double() / total_samples;
avg_loss = np.mean(losses)
199                 report = classification_report(all_labels, all_preds,
target_names=['Não Venda', 'Venda'], output_dict=True, zero_division
=0)
200                 return accuracy, avg_loss, report, all_labels, all_preds
201
202 # --- Etapa 6: Fine-tuning e Comparação dos Modelos ---
203 if not df_total.empty:
204     print("\n--- Etapa 6: Fine-tuning e Comparação BERT vs mBERT ---")
205
206     EPOCHS = 4
207     results = {}
208
209     try:
210         # --- Treinamento BERT ---
211         print("\n--- Iniciando Fine-tuning do BERT ('bert-base-uncased')
---")

```

```

212     model_bert = AutoModelForSequenceClassification.from_pretrained(
213         'bert-base-uncased', num_labels=2).to(device)
214     optimizer_bert = AdamW(model_bert.parameters(), lr=2e-5)
215     total_steps_bert = len(train_loader_bert) * EPOCHS
216     scheduler_bert = get_linear_schedule_with_warmup(optimizer_bert,
217         num_warmup_steps=0, num_training_steps=total_steps_bert)
218     best_val_accuracy_bert = 0
219
220     for epoch in range(EPOCHS):
221         print(f'\nEpoch {epoch + 1}/{EPOCHS}')
222         train_acc, train_loss = train_epoch(model_bert,
223             train_loader_bert, optimizer_bert, device, scheduler_bert)
224         val_acc, val_loss, _, _, _ = eval_model(model_bert,
225             val_loader_bert, device)
226         print(f'Train loss {train_loss:.4f} accuracy {train_acc:.4f}')
227
228         print(f'Val    loss {val_loss:.4f} accuracy {val_acc:.4f}')
229         if val_acc > best_val_accuracy_bert: best_val_accuracy_bert
230             = val_acc
231
232     results['BERT'] = {'model': model_bert, 'tokenizer':
233         tokenizer_bert, 'val_acc': best_val_accuracy_bert}
234
235     # --- Treinamento mBERT ---
236     print("\n--- Iniciando Fine-tuning do mBERT ('bert-base-
237         multilingual-cased') ---")
238
239     model_mbert = AutoModelForSequenceClassification.from_pretrained
240         ('bert-base-multilingual-cased', num_labels=2).to(device)
241     optimizer_mbert = AdamW(model_mbert.parameters(), lr=2e-5)
242     total_steps_mbert = len(train_loader_mbert) * EPOCHS
243     scheduler_mbert = get_linear_schedule_with_warmup(
244         optimizer_mbert, num_warmup_steps=0, num_training_steps=
245         total_steps_mbert)
246     best_val_accuracy_mbert = 0
247
248     for epoch in range(EPOCHS):
249         print(f'\nEpoch {epoch + 1}/{EPOCHS}')
250         train_acc, train_loss = train_epoch(model_mbert,
251             train_loader_mbert, optimizer_mbert, device, scheduler_mbert)
252         val_acc, val_loss, _, _, _ = eval_model(model_mbert,
253             val_loader_mbert, device)
254         print(f'Train loss {train_loss:.4f} accuracy {train_acc:.4f}')
255
256         print(f'Val    loss {val_loss:.4f} accuracy {val_acc:.4f}')
257         if val_acc > best_val_accuracy_mbert:
258             best_val_accuracy_mbert = val_acc
259
260     results['mBERT'] = {'model': model_mbert, 'tokenizer':
261         tokenizer_mbert, 'val_acc': best_val_accuracy_mbert}

```

```

243
244     except RuntimeError as e:
245         if "out of memory" in str(e):
246             print("\n\n" + "="*50)
247             print("ERRO: ESTOURO DE MEMÓRIA (CUDA out of memory).")
248             print("A RAM da GPU do Colab não foi suficiente para o
BATCH_SIZE atual.")
249             print("Tente reiniciar o ambiente e diminuir o BATCH_SIZE
para 2 ou 1 na Etapa 4.")
250             print("="*50 + "\n\n")
251             df_total = pd.DataFrame()
252         else:
253             raise e
254
255
256 # --- Etapa 7: Avaliação Final no Conjunto de Teste ---
257 if not df_total.empty and results:
258     print("\n\n--- Etapa 7: Avaliação Final no Conjunto de Teste ---")
259
260     # Dicionários para armazenar DataFrames
261     df_reports = {}
262
263     # Avaliar BERT no Teste
264     test_acc_bert, _, test_report_dict_bert, test_labels_bert,
test_preds_bert = eval_model(results['BERT']['model'],
test_loader_bert, device)
265     results['BERT']['test_acc'] = test_acc_bert
266     df_reports['BERT'] = pd.DataFrame(test_report_dict_bert).transpose()
267
268     # Avaliar mBERT no Teste
269     test_acc_mbert, _, test_report_dict_mbert, test_labels_mbert,
test_preds_mbert = eval_model(results['mBERT']['model'],
test_loader_mbert, device)
270     results['mBERT']['test_acc'] = test_acc_mbert
271     df_reports['mBERT'] = pd.DataFrame(test_report_dict_mbert).transpose()
272
273     print("\n\n--- Comparação Final ---")
274     print(f"Acurácia no Teste (BERT): {results['BERT']['test_acc']:.2%}"
)
275     print(f"Acurácia no Teste (mBERT): {results['mBERT']['test_acc
']:.2%}")
276
277     print("\nRelatório BERT:")
278     print(df_reports['BERT'])
279     print("\nRelatório mBERT:")
280     print(df_reports['mBERT'])

```

```

281
282     # Salvar Matrizes de Confusão
283     cm_bert = confusion_matrix(test_labels_bert, test_preds_bert)
284     plt.figure(figsize=(7, 5))
285     sns.heatmap(cm_bert, annot=True, fmt='d', cmap='Blues', xticklabels
286                 =['Previsto: Não Venda', 'Previsto: Venda'], yticklabels=['Real: Não
287                 Venda', 'Real: Venda'])
288     plt.title('Matriz de Confusão (BERT)')
289     plt.savefig('matriz_confusao_bert.png', dpi=150, bbox_inches='tight'
290               )
291     plt.show()
292
293     cm_mbert = confusion_matrix(test_labels_mbert, test_preds_mbert)
294     plt.figure(figsize=(7, 5))
295     sns.heatmap(cm_mbert, annot=True, fmt='d', cmap='Blues', xticklabels
296                 =['Previsto: Não Venda', 'Previsto: Venda'], yticklabels=['Real: Não
297                 Venda', 'Real: Venda'])
298     plt.title('Matriz de Confusão (mBERT)')
299     plt.savefig('matriz_confusao_mbert.png', dpi=150, bbox_inches='tight'
300               )
301     plt.show()
302
303     # --- NOVA ETAPA 8: Exportação dos Resultados ---
304     if not df_total.empty and 'df_reports' in locals():
305         print("\n--- Etapa 8: Gerando arquivos para download ---")
306
307         output_filename = 'Relatorio_Analise_BERT_mBERT.xlsx'
308
309         # Criar a planilha Excel com várias abas
310         with pd.ExcelWriter(output_filename, engine='openpyxl') as writer:
311             df_reports['BERT'].to_excel(writer, sheet_name='Relatorio_BERT')
312             df_reports['mBERT'].to_excel(writer, sheet_name='Relatorio_mBERT')
313
314         # Adicionar nota sobre insights
315         df_insights_info = pd.DataFrame({
316             'Nota': [
317                 'Insights de features (palavras/comportamentos) não são
318                 gerados por esta abordagem.',
319                 'Modelos Transformer (BERT/mBERT) são "caixas-pretas". A
320                 interpretação direta (como os coeficientes da V13) não é trivial.',
321                 'Técnicas avançadas de interpretação (SHAP) mostraram-se
322                 computacionalmente inviáveis no ambiente Colab (estouro de RAM).',
323                 'O foco desta abordagem é a performance preditiva máxima'
324             ]
325         })

```

```

317         df_insights_info.to_excel(writer, sheet_name='Sobre_Insights',
318                                     index=False)
319
320     print(f"Planilha de resultados '{output_filename}' e imagens das
321           matrizes prontas.")
322
323     # Acionar o download no Colab
324     try:
325         files.download('matriz_confusao_bert.png')
326         files.download('matriz_confusao_mbert.png')
327         files.download(output_filename)
328         print("Downloads iniciados...")
329     except Exception as e:
330         print(f"Erro ao tentar baixar os arquivos: {e}")
331         print("Você pode baixar os arquivos manualmente no painel à
332               esquerda (clique em 'Atualizar').")
333
334 else:
335     print("\nNão foi possível gerar os arquivos de resultado (
336           treinamento pode ter falhado por falta de RAM).")
337
338 # --- ETAPA 9: Análise de Conversas Exemplares (Adaptada para o melhor
339 # modelo) ---
340 if not df_total.empty and results:
341     print("\n\n" + "="*50 + "\nETAPA 9: ANÁLISE DE CONVERSAS EXEMPLARES
342           (com melhor modelo)\n" + "="*50)
343
344     # Escolher o melhor modelo baseado na acurácia do teste
345     best_model_name = 'mBERT' if results['mBERT']['test_acc'] > results[
346 'BERT']['test_acc'] else 'BERT'
347     best_model = results[best_model_name]['model']
348     best_tokenizer = results[best_model_name]['tokenizer']
349     print(f"Usando o melhor modelo: {best_model_name}")
350
351     def predict_single_text(text, model, tokenizer, device, max_len=
352 MAX_LEN):
353         model.eval()
354         processed_text = preprocessar_para_bert_completo(text)
355         encoding = tokenizer.encode_plus(
356             processed_text, add_special_tokens=True, max_length=max_len,
357             return_token_type_ids=False,
358             padding='max_length', truncation=True, return_attention_mask
359             =True, return_tensors='pt',
360         )
361         input_ids = encoding['input_ids'].to(device); attention_mask =
362         encoding['attention_mask'].to(device)

```

```

353     with torch.no_grad(): outputs = model(input_ids=input_ids,
354         attention_mask=attention_mask)
355     logits = outputs.logits; probs = torch.softmax(logits, dim=1)
356     prob_venda = probs[0][1].item(); prediction = torch.argmax(
357         logits, dim=1).item()
358     status = "PROMISSORA (Venda)" if prediction == 1 else "NÃO
359     PROMISSORA (Não Venda)"
360     return status, prob_venda
361
362     all_texts = df_total['texto_completo'].tolist()
363     all_probs_best = []
364     print(f"\nCalculando probabilidades com {best_model_name} para todos
365     os dados (pode levar um tempo)...")
366     for text in tqdm(all_texts):
367         _, prob = predict_single_text(text, best_model, best_tokenizer,
368             device)
369         all_probs_best.append(prob)
370
371     df_total['prob_venda_best_bert'] = all_probs_best
372     conversa_mais_promissora = df_total.loc[df_total['
373     prob_venda_best_bert'].idxmax()]
374     conversa_menos_promissora = df_total.loc[df_total['
375     prob_venda_best_bert'].idxmin()]
376
377     print(f"\n\ n    CONVERSA MAIS PROMISSORA (Segundo {best_model_name})
378     ")
379     print("-----")
380     print(f"Arquivo: {conversa_mais_promissora['arquivo']} |
381     Probabilidade de Venda: {conversa_mais_promissora['
382     prob_venda_best_bert']:.2%}")
383     # print("\n--- Texto Original ---"); print(conversa_mais_promissora
384     ['texto_completo'])
385     print(f"\n\ n    CONVERSA MENOS PROMISSORA (Segundo {best_model_name}
386     })")
387     print("-----")
388     print(f"Arquivo: {conversa_menos_promissora['arquivo']} |
389     Probabilidade de Venda: {conversa_menos_promissora['
390     prob_venda_best_bert']:.2%}")
391     # print("\n--- Texto Original ---"); print(conversa_menos_promissora
392     ['texto_completo'])

```