CENTRO UNIVERSITÁRIO UNIFECAF

3º semestre Gestão da Tecnologia da Informação

Machine Learning & Deep Learning

Estudo de caso

Classificação Inteligente de Chamados: Construção de um Modelo de Machine Learning para Priorização Automática no Atendimento ao Cliente

Raphael Henrique Silva Serafim 106488

TABOÃO DA SERRA/SP 2025

CENTRO UNIVERSITÁRIO UNIFECAF

SUMÁRIO

1 INTRODUÇÃO

Este projeto estabelece uma base técnica para um pipeline de Machine Learning (ML) focado na **priorização de chamados de clientes**. O objetivo central é classificar a `urgencia` dos chamados, que é o *target* gerado com base em regras de negócio

simuladas, onde um alto `dias_atraso` E um alto `valor_total_divida` resultam em "Alta Urgência". O projeto utiliza uma abordagem híbrida, combinando **dados textuais** (`texto_chamado`) e **dados tabulares** (financeiros). Dada a **natureza desbalanceada intrínseca** das classes, o projeto prioriza o **F1-Score Ponderado** como métrica de controle, considerada mais robusta que a Acurácia.

2 CONFIGURAÇÃO E GERAÇÃO DE DADOS

2.1 Configuração do Ambiente e Dependências

A fase de *bootstrap* do projeto envolve a instalação silenciosa de dependências de ponta a ponta, incluindo ferramentas para simulação de dados (`Faker`), modelos robustos (`xgboost`), técnicas de balanceamento (`imbalanced-learn`) e Deep Learning (`tensorflow`, `torch`). As bibliotecas são importadas de forma modular, abrangendo **Manipulação/Visualização** (`pandas`, `numpy`, `seaborn`), **Pré-processamento** (`TfidfVectorizer`, `LabelEncoder`, `StandardScaler`) e **Modelagem** (preparando `RandomForestClassifier`, `SVC`, `XGBClassifier` e `tensorflow.keras`).

2.2 Estratégia de Carregamento e Dados Sintéticos

A fonte primária dos dados utiliza uma **abordagem de *failover*** (contingência). A função `upload_data()` tenta carregar um arquivo CSV via *upload* interativo. Se o *upload* falhar, a função `criar_dados_simulados()` gera um dataset de 1000 registros sintéticos com plausibilidade de negócio. O `texto_chamado` é gerado ligado ao nível de urgência para introduzir a correlação necessária para o modelo de Processamento de Linguagem Natural (NLP).

3 ANÁLISE EXPLORATÓRIA DE DADOS E PRÉ-PROCESSAMENTO

3.1 Análise de Correlação e Estatísticas

Para o cálculo da **Matriz de Correlação**, variáveis categóricas (`urgencia` e `historico_pagamento`) são codificadas ordinalmente (Baixa=0, Média=1, Alta=2). A visualização via *heatmap* é utilizada para demonstrar a força e a direção da relação entre as variáveis numéricas e o *target* codificado. Espera-se que haja uma **alta correlação positiva** entre `dias_atraso`, `valor_total_divida` e `urgencia_encoded_corr`, o que valida a simulação das regras de negócio.

A **Análise por Grupos** (`groupby`) gera estatísticas descritivas (média, mediana, desvio-padrão) para as variáveis de risco por nível de urgência.

3.2 Pré-processamento de Dados Textuais (NLP)

A função `preprocess_text` é essencial para implementar a limpeza e a normalização dos dados textuais. O *pipeline* de limpeza inclui: **Conversão para minúsculas**, **Remoção de Pontuação/Símbolos** (mantendo apenas letras e espaços) e **Filtro de *Stop Words*** (palavras de alta frequência e baixo significado preditivo em português). Este processamento reduz o ruído e a dimensionalidade do vocabulário, preparando o texto para a vetorização TF-IDF.

3.3 Codificação do Target e Desbalanceamento

O **LabelEncoder** é usado para converter a variável categórica de resposta (`urgencia`) em um formato numérico (0, 1, 2). Um gráfico de barras confirma visualmente o **desbalanceamento intrínseco de classes**.

Esta visualização justifica a necessidade de usar o **F1-Score Ponderado** e técnicas de balanceamento como `class_weight`.

3.4 Validação do Sinal Preditivo (EDA Visual)

A Análise Exploratória de Dados (EDA) valida o sinal preditivo.

4 ENGENHARIA DE FEATURES E COMBINAÇÃO HÍBRIDA

4.1 Vetorização TF-IDF e Análise de Top Features

A vetorização do texto limpo é executada pelo **TfidfVectorizer**, que converte o texto em uma matriz numérica esparsa. Parâmetros sêniores foram definidos para otimização: `ngram_range=(1, 2)`, `max_features=500`, `min_df=2` e `max_df=0.8`.

A **Análise das Top Features** exibe os 15 termos com a maior Soma dos Pesos TF-IDF, revelando quais termos o modelo utilizará como preditores mais fortes de urgência.

4.2 Fusão e Normalização de Features Múltiplas

O tratamento e a **Codificação Ordinal** da *feature* `historico_pagamento` preservam sua ordem de importância. Os histogramas são plotados para visualizar a distribuição dos dados antes da normalização.

O **StandardScaler** é ajustado apenas nas *features* contínuas (`dívida`, `atraso`), sendo fundamental que a *feature* ordinal não seja escalada. A combinação das *features* é feita usando `np.hstack`, gerando a matriz final `X_combined`.

4.3 Divisão Estratificada do Dataset

A divisão final dos dados utiliza o `train_test_split` com o parâmetro **stratify=y**. Esta decisão garante que a proporção das classes (Alta, Média, Baixa) seja preservada tanto no conjunto de treino quanto no de teste.

Para a utilização de Redes Neurais, a matriz esparsa `X_combined` é convertida para matrizes densas (`.toarray()`), e o *target* (y) é convertido para o formato **One-Hot Encoding**.

5 MODELAGEM E AVALIAÇÃO DE DESEMPENHO

5.1 Treinamento de Modelos Clássicos

Três arquiteturas clássicas distintas são comparadas: **Random Forest (RF)**, **XGBoost** e **Support Vector Machine (SVM)**. Para mitigar o desbalanceamento, os modelos RF e SVM são configurados com o parâmetro **class_weight='balanced'**. O **F1-Score Ponderado** (`average='weighted'`) é a métrica principal, e a **Matriz de Confusão** é gerada para diagnosticar Falsos Negativos e Falsos Positivos.

5.2 Treinamento de Rede Neural (Deep Learning)

Uma arquitetura **Sequential** de Deep Learning foi introduzida. Incluiu camadas ocultas *Dense* com ativação **ReLU** e **Dropout(0.5)** para regularização. O modelo é compilado usando o otimizador **Adam** e a função de perda **categorical_crossentropy**.

5.3 Análise de Erros e Falsos Negativos

Esta etapa crucial foca na gestão de risco. O principal objetivo é isolar e quantificar os **Falsos Negativos**. Um DataFrame (`erros_df`) é criado filtrando apenas os registros onde o Verdadeiro é diferente do Predito.

5.4 Análise de Interpretabilidade (Vetores de Suporte)

Para o modelo SVM, é realizada uma análise de interpretabilidade utilizando o conceito de **Vetores de Suporte (VS)**. A inspeção dos exemplos de chamados que são VS (exibindo texto, urgência, atraso) revela o perfil de cliente de fronteira.

A **Análise do Perfil dos Vetores de Suporte (VS)** visualiza a distribuição de `historico_pagamento` ou `dias_atraso` entre os VS e o dataset completo.

[CÉLULA 13] 🔎 (Insira o print dos gráficos que comparam o perfil de historico_pagamento e dias_atraso entre os Vetores de Suporte e o dataset total)

5.5 Comparação Final de Desempenho e Curvas de Aprendizado

A **Comparação Final de Desempenho** utiliza um *barplot* horizontal comparando o **F1-Score Ponderado** de todos os modelos. O ajuste do eixo X é utilizado para ampliar as diferenças entre *scores* altos.

A **Análise de Diagnóstico Avançado** (Curvas de Aprendizado) avalia a estabilidade e capacidade de generalização dos modelos em função do volume de dados.

6 DEPLOYMENT E SIMULAÇÃO OPERACIONAL

6.1 Persistência de Artefatos

O **RandomForestClassifier** (assumido como vencedor) é serializado. Quatro artefatos essenciais são persistidos usando `joblib.dump`: o modelo, o `tfidf_vectorizer.pkl`, o `scaler.pkl` e o `label_encoder.pkl`. A serialização de todos os pré-processadores garante a integridade e a ausência de *data leakage*.

6.2 Simulação e Teste de Unidade

A função `priorizar_chamado` simula o *pipeline* de produção, replicando a **exata sequência de pré-processamento**. Um **Teste de Unidade Funcional** é executado para validar o comportamento do modelo em cenários específicos (Alta, Média, Baixa Urgência).

6.3 Aplicação do Modelo no Dataset Completo

O processo simula um ambiente de produção em lote (*batch*). O código garante a **replicação fiel do *pipeline***, usando o método `.transform` em todos os pré-processadores salvos. O resultado principal é a criação da nova coluna `merged_df['urgencia_prevista']`.

7 BUSINESS INTELLIGENCE (BI) E ANÁLISES ESTRATÉGICAS

7.1 Geração do Dashboard de Monitoramento

O projeto finaliza com a entrega de valor para o negócio por meio de um **dashboard 2x2**. O dashboard sintetiza a performance do modelo e fornece *insights* operacionais.

O dashboard inclui: **Gráfico Real vs. Prevista**, **Precisão por Classe**, **Valor Médio da Dívida Prevista** e **Matriz de Confusão**.

7.2 Análise de Segmentação (Geográfica e Operacional)

7.2.1 Análise Geográfica

A **Análise Geográfica de Urgência Prevista** segmenta as previsões por estado. Essa visualização permite identificar **Focos de Risco** e **Otimizar o Call Center** geograficamente.

O **Mapa Coroplético** representa o ponto máximo da visualização, traduzindo a priorização em um mapa interativo.

A **Visualização Geoespacial da Densidade Total de Chamados** mapeia o volume absoluto de chamados por estado.

7.2.2 Análise Operacional

A **Análise de Segmentação Operacional** segmenta os resultados por `tipo_cliente` e `canal_contato`.

A **Análise do Impacto de Tentativas de Contato** correlaciona as métricas de risco com a eficiência da operação de cobrança (`tentativas_contato`).

8 CONCLUSÃO

O projeto implementou um *pipeline* completo de Machine Learning (ML) para priorização de chamados de urgência, desde a configuração do ambiente até a geração de *insights* de Business Intelligence. Foi comprovado que as *features* numéricas (`dias_atraso` e `valor_total_divida`) possuem correlação direta e lógica com a prioridade de atendimento. A metodologia adotada priorizou a robustez frente ao desbalanceamento de classes, utilizando a **estratificação**, o parâmetro **class_weight='balanced'** e o **F1-Score Ponderado** como métrica de decisão. O modelo vencedor foi persistido juntamente com todos os artefatos de pré-processamento, garantindo a integridade para o *deployment*. O resultado final fornece ao negócio uma classificação de prioridade de atendimento (`urgencia_prevista`), permitindo a otimização de recursos e a mitigação de Falsos Negativos (erros críticos de risco).