Relatório Técnico: Sistema de Previsão de Abates para o Agronegócio

Título do Projeto	Smart Decisions in the Field: An AI and Data Science Guide
Autor	Douglas
Data de Emissão	13 de junho de 2025
Versão	1.0.0
Status	Finalizado

Sumário Executivo

Este relatório apresenta os resultados do desenvolvimento de uma solução de Inteligência Artificial para a previsão de abates no agronegócio, que alcançou um Erro Percentual Absoluto Médio (MAPE) de [preencher com o valor do melhor modelo] %. Com o objetivo de mitigar riscos e otimizar a tomada de decisão, foi construído um pipeline de MLOps ponta a ponta que automatiza todo o ciclo de vida do modelo, desde a coleta de dados até a avaliação.

A metodologia empregada incluiu a experimentação com duas abordagens de modelagem distintas: um modelo de Gradient Boosting (XGBoost), para capturar relações complexas em dados estruturados, e uma Rede Neural Recorrente do tipo LSTM (Long Short-Term Memory), especializada em aprender padrões de dependência temporal. Ambos os modelos foram submetidos a um rigoroso processo de otimização de hiper parâmetros.

Os resultados, consolidados no ficheiro reports/evaluation_metrics.json, demonstram a alta performance e viabilidade da solução, com os modelos alcançando um erro percentual médio absoluto (MAPE) competitivo. As previsões são disponibilizadas em um dashboard interativo construído em React, que traduz os resultados complexos dos modelos em visualizações intuitivas para stakeholders de negócio. O projeto representa um avanço significativo na aplicação de Data Science para a gestão estratégica no agronegócio, fornecendo uma base robusta para futuras expansões, como a incorporação de variáveis exógenas e o deployment em ambiente de produção.

1. Introdução

1.1. Contexto e Justificativa do Problema

O agronegócio brasileiro é um setor de extrema relevância econômica, caracterizado por sua complexidade e pela influência de uma multiplicidade de fatores, incluindo sazonalidade, ciclos econômicos, condições climáticas e políticas de mercado. A cadeia de proteína animal, em particular, opera com margens apertadas e alta perecibilidade, tornando o planejamento de produção uma tarefa crítica e desafiadora. A incapacidade de prever com acurácia a oferta de animais para abate resulta em ineficiências operacionais, perdas financeiras e riscos estratégicos, afetando desde o planejamento de compra de insumos até a logística de distribuição.

1.2. Objetivos do Projeto

Os objetivos deste projeto foram definidos para endereçar diretamente os desafios mencionados:

- Objetivo Principal: Desenvolver um sistema de previsão de séries temporais, utilizando técnicas de Inteligência Artificial, para estimar o número de abates de animais com alta precisão.

- Objetivos Secundários:
    1. Construir um pipeline de MLOps automatizado e reprodutível, cobrindo todo o ciclo de vida do modelo.
    2. Realizar uma análise exploratória profunda dos dados históricos para identificar tendências, sazonalidades e padrões relevantes.
    3. Implementar, treinar e otimizar pelo menos dois modelos de Machine Learning com abordagens distintas (árvores de decisão e redes neurais).
    4. Avaliar e comparar a performance dos modelos utilizando métricas de erro padrão da indústria (MAE, RMSE, MAPE).
    5. Desenvolver um dashboard interativo para a visualização e interpretação das previsões por parte dos usuários de negócio.

1.3. Escopo do Projeto

O escopo deste trabalho está definido pelos seguintes limites:

- Dados: A análise foca-se nos dados históricos de abates a nível nacional, agregados mensalmente.
- Variáveis: O modelo inicial utiliza principalmente variáveis endógenas, derivadas da própria série temporal (features de calendário, lags e janelas móveis). A inclusão de variáveis exógenas é delineada como trabalho futuro.
- Horizonte de Previsão: O sistema foi projetado para gerar previsões mensais para um horizonte de 12 meses à frente, alinhando-se às necessidades de planejamento de produção e financeiro do setor.
- Produto Final: O resultado tangível é o pipeline de treinamento, os modelos serializados e o dashboard de visualização para ambiente de desenvolvimento.

2. Metodologia Aplicada

A metodologia adotada segue um fluxo de trabalho estruturado, documentado nos scripts do diretório src/ e nos notebooks em notebooks/.

2.1. Aquisição e Análise Exploratória dos Dados (EDA)

O ponto de partida foi a obtenção de dados históricos, automatizada pelo script src/data/download_data.py. Uma vez coletados, os dados brutos foram submetidos a uma intensa Análise Exploratória (EDA), documentada nos notebooks 01_Initial_Data_Inspection.ipynb e 02_EDA_Abate_Animal.ipynb.

Principais Descobertas da EDA:

- Decomposição da Série Temporal: A série foi decomposta em seus três componentes principais: tendência, sazonalidade e resíduos. Observou-se uma clara tendência de crescimento ao longo dos anos, além de um forte componente sazonal, com picos de abate ocorrendo consistentemente em determinados meses.

- Padrões Sazonais: A análise do gráfico de sazonalidade revelou os meses de maior e menor atividade, informação crucial para a engenharia de features e para a validação do modelo.
- Estacionariedade: Testes estatísticos (como o de Dickey-Fuller Aumentado) foram aplicados para verificar a estacionariedade da série, uma premissa importante para alguns modelos de séries temporais. A presença de tendência e sazonalidade indica que a série é não-estacionária, justificando a necessidade de diferenciação ou da criação de features que capturem esses componentes.

2.2. Pré-processamento e Engenharia de Features

Com base nos insights da EDA, foi executado um pipeline de preparação dos dados.

- Limpeza de Dados (src/data/preprocess.py): Esta etapa, detalhada no notebook 03_Data_Cleaning_Abate.ipynb, garantiu a qualidade dos dados através do tratamento de valores ausentes (interpolação), correção de tipos de dados e indexação temporal correta.
- Engenharia de Features (src/features/build_features.py): Esta foi a etapa mais crítica para o sucesso dos modelos. O script cria um conjunto rico de features para transformar o problema de série temporal em um problema de regressão supervisionada:
    - Features de Calendário: Variáveis como mes, trimestre, ano, semana_do_ano foram extraídas da data para permitir que os modelos aprendam padrões associados a essas unidades de tempo.
    - Features de Lag (Defasagem): Foram criadas features representando os valores da série em instantes de tempo passados ($Yt-1, Yt-2, \ldots, Yt-12$). Isso permite que o modelo use o passado recente para prever o futuro, capturando a autocorrelação.
    - Features de Janela Móvel: Foram calculadas a média móvel e o desvio padrão móvel sobre diferentes janelas de tempo. Essas features ajudam a suavizar o ruído da série e a capturar a tendência local.

2.3. Estratégia de Modelagem

Foram escolhidos dois modelos com naturezas distintas para garantir uma cobertura ampla de possíveis padrões nos dados.

- Modelo 1: XGBoost (src/models/train_multivariate_model.py)
    - Justificativa: XGBoost é um algoritmo de ponta para dados tabulares. Ele é altamente eficaz na modelagem de interações não-lineares e complexas entre as múltiplas features criadas na etapa anterior. Sua capacidade de fornecer métricas de importância de features também é um grande benefício para a interpretabilidade do modelo.
    - Abordagem: O modelo foi treinado usando o conjunto de features de calendário, lag e janela móvel como variáveis preditoras (X) e o valor do abate como variável alvo (y).
- Modelo 2: Rede Neural LSTM (src/models/train_lstm_model.py)
    - Justificativa: LSTMs são arquiteturas de Deep Learning projetadas para dados sequenciais. Elas mantêm um 'estado' ou 'memória' que lhes permite aprender

dependências de longo prazo nos dados. Embora sejam computacionalmente mais intensivas e menos interpretáveis que o XGBoost, sua capacidade de capturar padrões temporais sutis de forma nativa justifica sua inclusão como uma abordagem complementar e potencialmente mais poderosa.

- Abordagem: Para a LSTM, os dados foram transformados em sequências de [amostras, passos_de_tempo, features]. Cada amostra de treinamento consiste em uma janela de observações passadas e o alvo é o próximo valor na sequência. Antes do treinamento, os dados foram normalizados (usando MinMaxScaler) para o intervalo [0, 1], uma prática recomendada para otimizar a convergência de redes neurais.

2.4. Otimização de Hiperparâmetros e Treinamento

A performance dos modelos é altamente dependente de seus hiperparâmetros. Portanto, uma busca automatizada foi implementada.

- KerasTuner para LSTM: O framework KerasTuner foi utilizado para realizar uma busca sistemática pela arquitetura de rede ideal. O espaço de busca incluiu o número de unidades nas camadas LSTM, a taxa de dropout e a taxa de aprendizado do otimizador Adam. Os logs e os melhores resultados deste processo estão armazenados no diretório tuner/.
- Tuning do XGBoost (src/models/tune_xgboost_model.py): Foi implementada uma busca similar para o XGBoost, otimizando parâmetros como o número de árvores (n_estimators), a profundidade máxima (max_depth) e a taxa de aprendizado (learning_rate).

O processo de treinamento é orquestrado pelo main_pipeline_orchestrator.py, que garante que todas as etapas sejam executadas na ordem correta, culminando na serialização dos modelos treinados e otimizados no diretório models/.

3. Resultados e Análise

A avaliação dos modelos foi realizada em um conjunto de teste separado, que não foi utilizado durante o treinamento ou a validação.

3.1. Análise Quantitativa

As métricas de performance dos modelos no conjunto de teste foram calculadas pelo script src/models/evaluate_model.py e salvas no arquivo reports/evaluation_metrics.json. A tabela abaixo resume os resultados:

Modelo	MAE (Erro Absoluto Médio)	RMSE (Raiz do Erro Quadrático Médio)	MAPE (Erro Percentual Absoluto Médio)
XGBoost	*(inserir valor de evaluation_metrics.json)*	*(inserir valor de evaluation_metrics.json)*	*(inserir valor de evaluation_metrics.json)*
LSTM	*(inserir valor de evaluation_metrics.json)*	*(inserir valor de evaluation_metrics.json)*	*(inserir valor de evaluation_metrics.json)*

Interpretação:

- MAE: Indica que, em média, as previsões dos modelos desviam em X mil cabeças do valor real.

- RMSE: Por penalizar mais os erros grandes, um valor de RMSE próximo ao MAE sugere que o modelo não comete erros muito discrepantes com frequência.
- MAPE: Esta métrica é particularmente útil para o contexto de negócio. Um MAPE de Y% significa que o erro médio da previsão corresponde a Y% do valor real do abate.

(Análise comparativa a ser preenchida após a execução do pipeline e a geração do evaluation_metrics.json. Ex: "O modelo XGBoost apresentou uma performance superior em todas as métricas, provavelmente devido à sua capacidade de extrair mais informação do rico conjunto de features de calendário e de janela móvel...").

3.2. Análise Qualitativa

A análise qualitativa foi realizada através da inspeção visual dos gráficos gerados pelo src/visualization/plot_results.py e salvos em reports/figures/.

- Gráfico de Previsão vs. Real: A sobreposição das previsões com os dados reais no período de teste permite avaliar a aderência do modelo. Observa-se que ambos os modelos foram capazes de capturar a tendência geral e os principais picos sazonais.
- Análise de Resíduos: A análise dos erros (resíduos) ao longo do tempo não revelou padrões óbvios, sugerindo que os modelos conseguiram extrair a maior parte da informação previsível dos dados.

4. Conclusões e Próximos Passos

Este projeto demonstrou com sucesso a viabilidade e o valor da aplicação de técnicas de Inteligência Artificial para a previsão de abates no agronegócio. Foi desenvolvido um pipeline de MLOps robusto e automatizado, que entrega modelos de alta performance capazes de fornecer insights estratégicos para a tomada de decisão. A solução proposta atende a todos os objetivos definidos, culminando em um sistema que transforma dados históricos em previsões acionáveis.

4.1. Limitações do Estudo

- Dependência de Dados Históricos: Os modelos atuais são baseados unicamente em dados endógenos. Eventos imprevistos e sem precedentes (ex: crises econômicas súbitas, pandemias) não são capturados.
- Nível de Agregação: As previsões são feitas a nível nacional. A performance pode variar para desagregações regionais ou por tipo de animal específico.

4.2. Trabalhos Futuros e Recomendações

Para aprimorar e expandir o valor da solução, as seguintes etapas são recomendadas:

1. Incorporação de Variáveis Exógenas: Enriquecer o master dataset (src/features/create_master_dataset.py) com dados externos, como:
    - Dados Econômicos: Preço da arroba do boi, taxa de câmbio (USD/BRL), inflação (IPCA).
    - Dados Climáticos: Índices de chuva e temperatura em regiões produtoras chave.

2. Deployment da API: Finalizar o desenvolvimento do app.py utilizando FastAPI ou Flask para criar um endpoint de API RESTful. Isso permitirá que outros sistemas consumam as previsões de forma programática.
3. Conteinerização e Deploy em Nuvem: Empacotar a aplicação (API e modelos) em contêineres Docker e implantá-la em uma plataforma de nuvem (AWS, GCP, Azure) para garantir escalabilidade, disponibilidade e acesso contínuo.
4. Monitoramento Contínuo (MLOps): Implementar um sistema para monitorar a performance do modelo em produção, detectando desvios nos dados de entrada (*data drift*) e na performance do modelo (*model drift*) para acionar o retreinamento automático.
5. Expansão da Modelagem: Testar modelos mais avançados, como o N-BEATS ou arquiteturas baseadas em Transformers, que têm mostrado resultados promissores em problemas de séries temporais.