

Modelo de previsão de views do Wikipedia

Gabriel Rosa, Carlos W, Arthur H, Kauan, Jefferson e Otavio

1. Visão Geral: Pipeline Modular de Forecasting

O projeto seguiu um fluxo estrito de 5 etapas, distribuídas em notebooks separados para garantir **modularidade** e **manutenibilidade**:

01

Exploração

Análise de anomalias (Outliers, Missing Values), Tendência e Sazonalidade (STL Decomposition).

02

Preparação

Limpeza, Normalização por série e Engenharia de Features (34+ features).

03

Modelagem

Implementação e treinamento de 12 modelos distintos (Estatísticos, ML, Deep Learning).

04

Avaliação

Comparação multi-métrica e Análise de Resíduos.

2. Transformação de Dados Crítica: Wide → Long Format

O dado original estava em **formato wide** (colunas = datas), inadequado para a maioria das bibliotecas de ML e de séries temporais. A primeira decisão técnica foi a transformação para o **formato long** (tidy data), usando a função melt.

Implementação

```
df_melted = df_raw.melt(  
    id_vars=['Page'],  
    var_name='Date',  
    value_name='Views'  
)
```

Justificativa

O formato longo facilita operações de agregação (groupby) e é o padrão para modelagem de séries temporais, garantindo compatibilidade e eficiência.



3. Feature Engineering I: Codificação Cíclica (Sazonalidade)

Para que modelos de ML entendam corretamente a natureza cíclica do tempo (ex: Dezembro e Janeiro são consecutivos), foi aplicada a transformação **Seno/Cosseno** em features como Month e DayOfWeek.

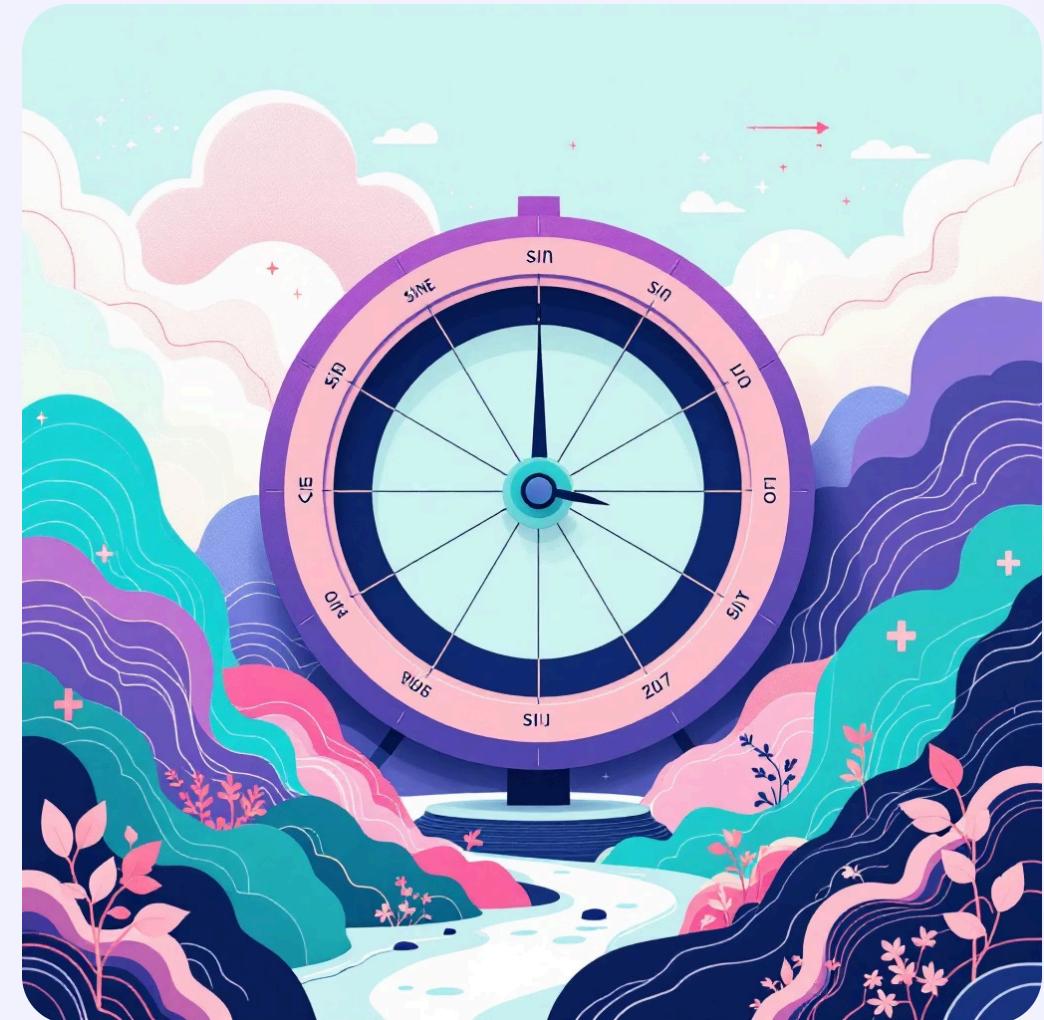
Fórmula Exemplo

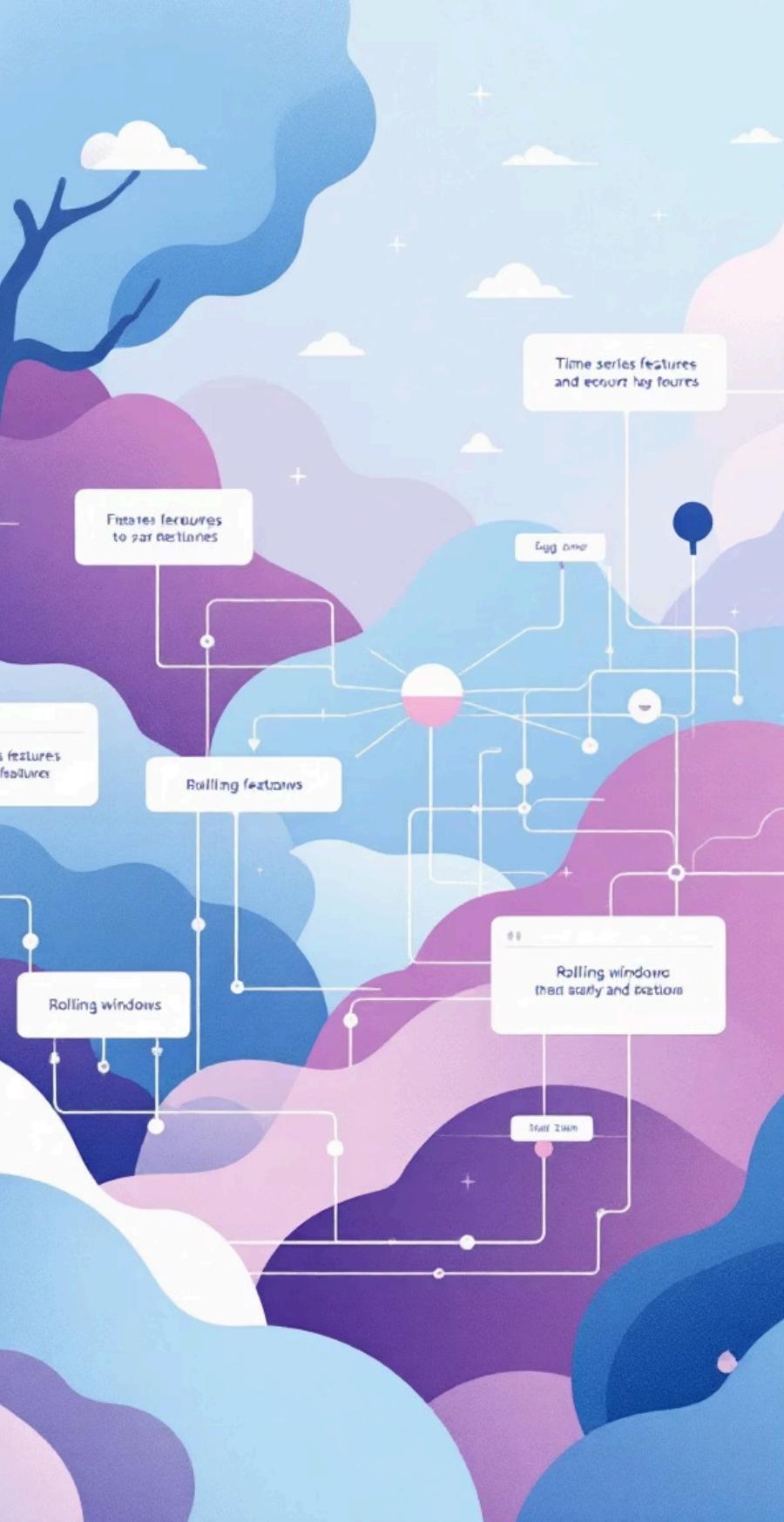
(Dia da Semana, ciclo de 7 dias):

$$X_{\sin} = \sin\left(\frac{2\pi \cdot X}{7}\right) \quad \text{e} \quad X_{\cos} = \cos\left(\frac{2\pi \cdot X}{7}\right)$$

Vantagem

Representa o tempo em um espaço de 2D, onde o início e o fim do ciclo (e.g., dia 1 e dia 7) ficam adjacentes, melhorando a performance em modelos como Redes Neurais e Regressões.





4. Feature Engineering II: Lags e Rolling Windows

As features mais importantes para a previsão foram criadas a partir da própria série temporal, totalizando mais de **34 features**.



Features de Lag

Valores defasados que capturam a autocorrelação da série, como `df['Views_lag_7'] = df['Views'].shift(7)` para capturar padrões semanais.



Features de Rolling Window

Estatísticas móveis que suavizam o ruído e capturam mudanças de regime, por exemplo, médias e desvios padrão móveis.



Tratamento de Outliers

Utilização de **Winsorização** (limites nos percentis 5 e 95) para preservar a continuidade temporal da série, ao invés de remover dados.

5. 🧑‍🔬 Estratégia de Pré-processamento: Normalização por Série

A decisão técnica crucial na normalização foi **escalar cada série (Page) de visualizações de forma independente.**



Implementação (Conceito)

```
# Correto: Escalar cada série independentemente  
for page in df['Page'].unique():  
    # Aplica scaler.fit_transform(page_data)  
    ...
```

Justificativa

Páginas muito populares (milhões de views) possuem escalas de magnitude totalmente diferentes de páginas de nicho. A normalização global faria com que as variações das séries menores fossem **esmagadas**, perdendo informação valiosa.

6. ⏳ Divisão de Dados: Time Series Split (Anti-Data Leakage)

A divisão dos dados foi **estritamente cronológica** (Treino 60% → Validação 20% → Teste 20%).

Princípio Fundamental

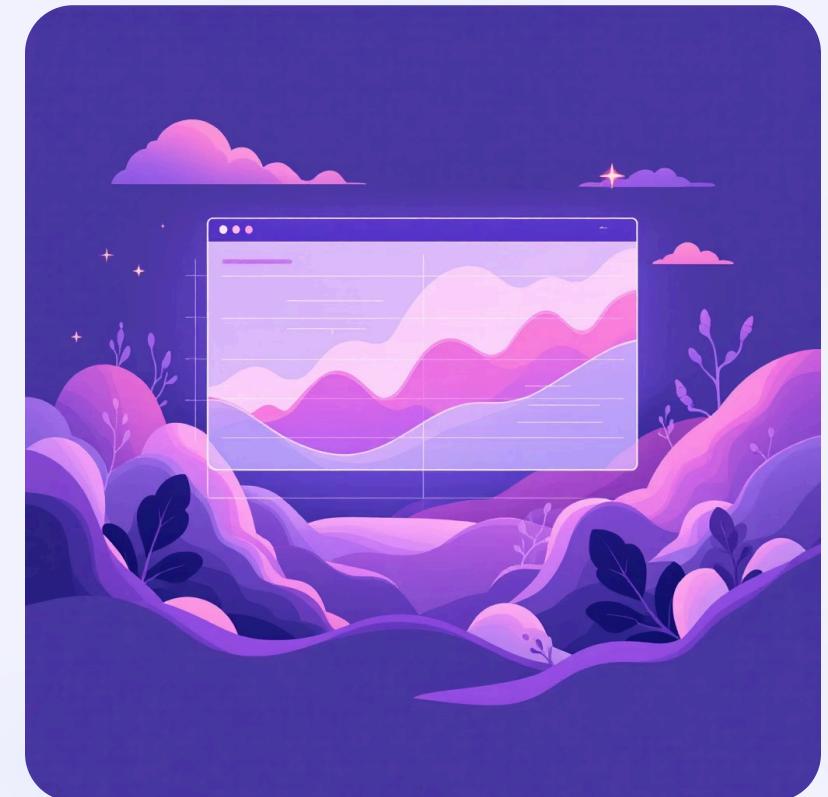
Nunca usar dados futuros para treinar!

Validação Cruzada

Foi utilizado o **TimeSeriesSplit** (Expanding Window) para simular o uso real e avaliar a **estabilidade temporal** do modelo, diferente do **KFold shuffle** que misturaria passado e futuro.

Exemplo do TimeSeriesSplit

```
from sklearn.model_selection import TimeSeriesSplit  
tscv = TimeSeriesSplit(n_splits=5)  
# Garante que train_idx sempre < val_idx
```



7. Modelos Estatísticos: ARIMA vs. Holt-Winters

Modelos base foram usados para estabelecer benchmarks e garantir interpretabilidade.



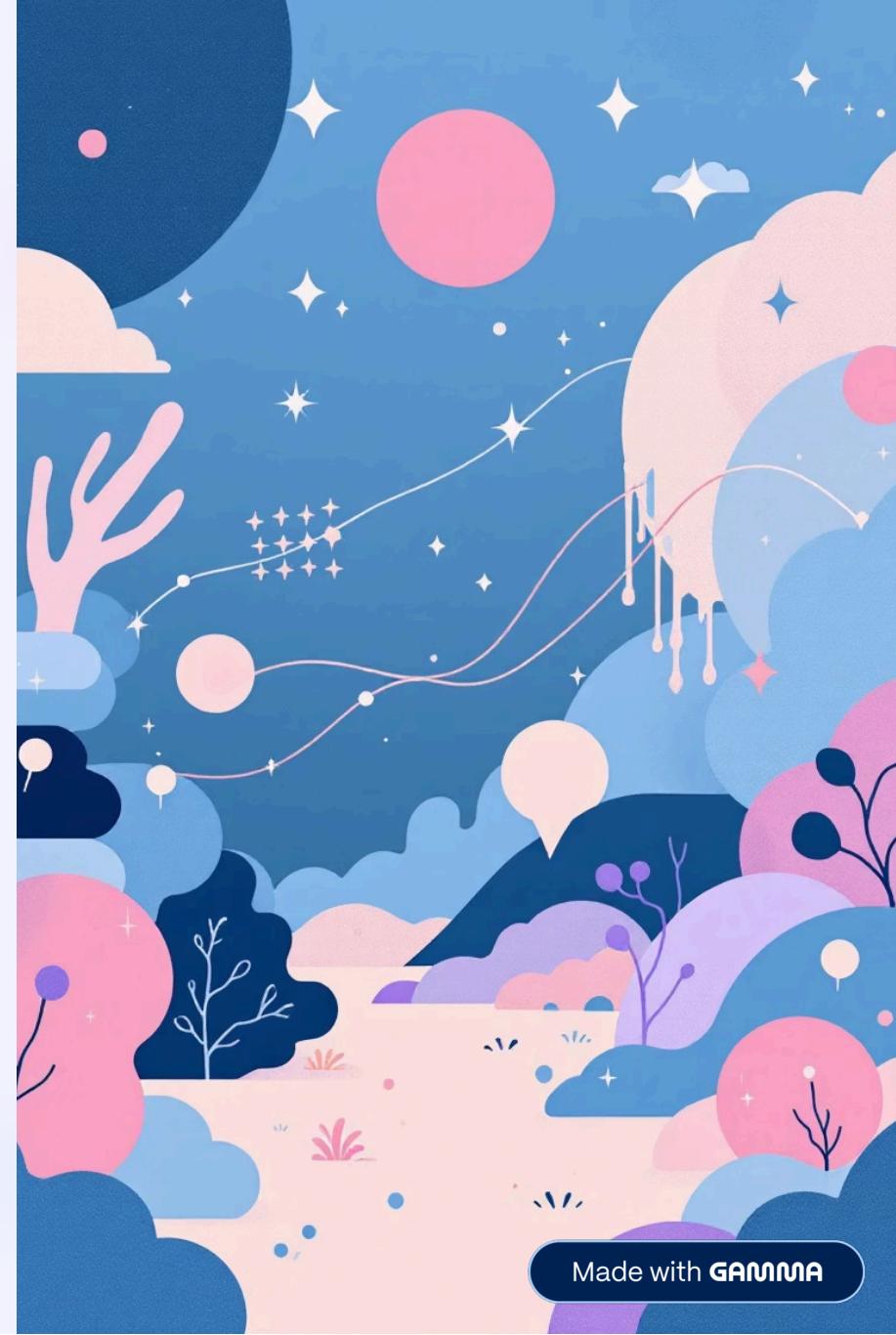
SARIMA

Estendido do ARIMA para capturar sazonalidade (e.g., semanal, $m=7$). Utilizou-se `pmdarima.auto_arima` para otimização automática dos parâmetros $(p, d, q)(P, D, Q)m$ com base no critério AIC.



Holt-Winters

Triple Exponential Smoothing, modela explicitamente e de forma interpretável os três componentes da série com otimização dos parâmetros de suavização: **Nível** (α), **Tendência** (β), **Sazonalidade** (γ).



8. 🤖 Modelos ML/Deep Learning: XGBoost e LSTM

As técnicas de ponta foram implementadas para capturar não-linearidades e dependências complexas.

XGBoost

1

Algoritmo de **Gradient Boosting** otimizado. Superior em dados tabulares, alavancou a performance preditiva através das **34+ features** criadas.

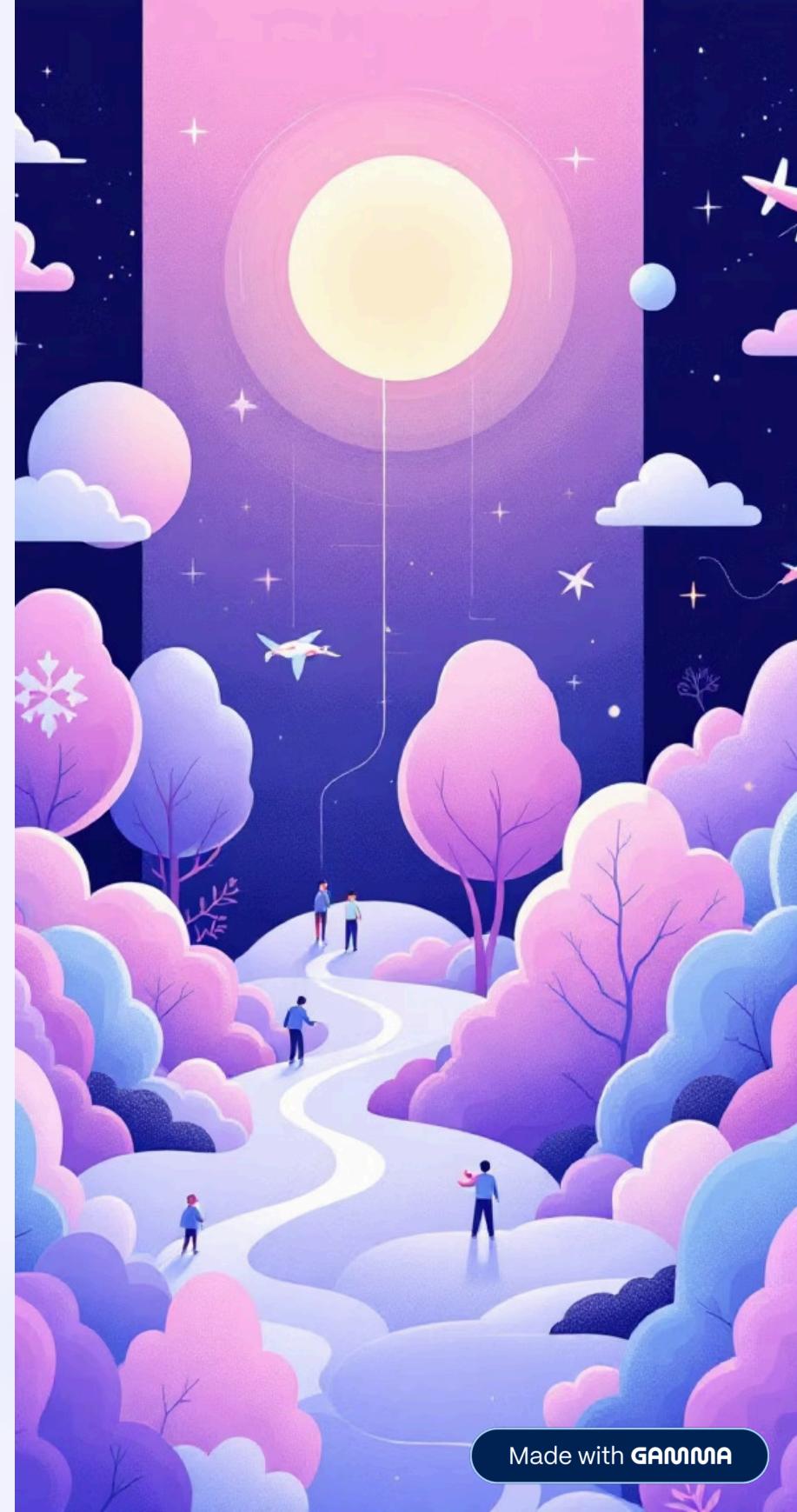
- Rapidez
- Regularização built-in
- Alto poder preditivo

LSTM

2

Long Short-Term Memory, uma Rede Neural Recorrente. Exige um reshape específico para a entrada: ([samples, timesteps, features](#)).

- Captura dependências de longo prazo (dias/semanas)
- Processamento automático de sequências



9. Métricas de Performance: Hierarquia e Interpretação

Foram utilizadas 4 métricas, com o **MAE** como principal critério de seleção devido à sua robustez a outliers e à sua **fácil interpretabilidade**.

Primária: MAE

(Mean Absolute Error).

Representa o erro médio na unidade original (número de visualizações).

Secundárias: RMSE e R²

RMSE (penaliza erros grandes) e **R²** (proporção da variância explicada).

Terciária: MAPE

(Mean Absolute Percentage Error). Erro percentual, útil para comparação entre páginas de diferentes escalas.

Um **MAPE < 10%** é considerado "Excelente".



10. Análise de Resíduos e Seleção Final

A seleção final do modelo baseou-se não apenas no melhor MAE, mas também na **estabilidade temporal** e na **qualidade dos resíduos**.

Análise de Resíduos

Os resíduos do modelo vencedor foram verificados para:

- **Normalidade** (Q-Q Plot, Teste Shapiro-Wilk).
- **Ausência de Autocorrelação** (ACF dos Resíduos, **Teste Ljung-Box**).

Resultado do Ljung-Box

`p_value > 0.05` indica **ruído branco** (bom), confirmando que o modelo capturou toda a estrutura temporal significativa.



Modelo Selecionado

XGBoost. Justificativa: Melhor MAE (melhoria de 35% sobre o Naive) e Resíduos com Ruído Branco, indicando alta capacidade preditiva e especificação robusta.