ESCOLA E FACULDADE DE TECNOLOGIA SENAI GASPAR RICARDO JÚNIOR
DESENVOLVIMENTO DE SISTEMAS

ANDRESSA ALMEIDA BARTOLOMEU CUSTÓDIO
MARCELA MORAES MULATO
NATÁLIA NOGUEIRA
RODRIGO FERREIRA SOARES

BIBLIOTECA

Projeto final de ciência de dados

PROFESSOR RESPONSÁVEL: ANDRÉ SOUZA
DISCIPLINA: CIÊNCIA DE DADOS

SOROCABA
20/06/2025

SUMÁRIO

1 INTRODUÇÃO

Este projeto tem como objeto de estudo o banco de dados disponibilizado publicamente pela Biblioteca da Universidade Federal de São Paulo (UNIFESP), o qual possui informações referentes ao acervo, empréstimos e perfis de usuários. O banco de dados escolhido é um exemplo real, o que possibilita uma ampla análise de seus dados por meio da Ciência de Dados.

O principal objetivo da análise presente nesse documento é aplicar as técnicas aprendidas durante as aulas de Ciência de Dados para identificar padrões de comportamento no uso da biblioteca. Para isso, foram utilizadas medidas de tendência central e dispersão, testes de normalidade, regressão linear simples, séries temporais e análise de dados categóricos. Os métodos citados permitiram compreender diferentes aspectos do funcionamento da biblioteca, desde a frequência de empréstimos ao longo do tempo até características associadas ao perfil dos usuários.

2 REFERENCIAL TEÓRICO

A estatística é a ciência que coleta, organiza, resume, analisa e interpreta dados para que determinada decisão possa ser tomada diante de uma proposta ou problemática. Para o desenvolvimento desse projeto, os conceitos estatísticos utilizados foram as medidas de tendência central e dispersão, testes de normalidade, regressão linear simples, séries temporais e análise de dados categóricos.

As medidas de tendência central são aquelas utilizadas para identificar pontos de concentração e visões gerais do comportamento de um conjunto de dados. Durante a análise da biblioteca, foram realizados cálculos de média, mediana e moda, que são medidas de tendência central, para analisar a quantidade de grupos de empréstimo que tiveram a mesma duração de tempo. Também foi analisada a variabilidade dos dados entre duração e quantidade de dias emprestados utilizando amplitude, variância e desvio padrão, que são medidas de dispersão. Além das medidas citadas, foram realizados os testes de normalidade, que verificam se a distribuição dos dados relacionados à quantidade de grupos de empréstimo que tiveram a mesma duração é normal, Shapiro-Wilk, geralmente utilizado em amostras pequenas, Komogorov-Smirnov, que compara a distribuição da amostra com uma distribuição normal, e Anderson-Darling, que verifica se os dados seguem a distribuição normal, como Komogorov-Smirnov, porém com maior sensibilidade a discrepâncias nos valores maiores e menores.

A regressão linear simples foi aplicada com o objetivo de encontrar a reta que melhor se ajusta aos dados da amostra entre o tempo do empréstimo e a quantidade de livros emprestados, minimizando o erro de diferença entre os valores reais e previstos, realizando uma previsão mais segura sobre os dados. Já as séries temporais observam os padrões ao longo do tempo, e nesse trabalho, foi analisada a relação de empréstimos realizados por dia. Foi analisada a presença de tendências e ciclos, e a decomposição da série temporal, observando também a sazonalidade e os ruídos. Por fim, foi realizada também a análise de dados categóricos, relacionando a quantidade de tipos de livros conforme o tipo de usuário através de um gráfico de barras e um gráfico de pizza. Todas as etapas do processo geraram visualizações com gráficos estatísticos como histogramas, boxplots e de dispersão.

As bibliotecas Python utilizadas para a realização dessas análises foram Pandas Python, que realizou a leitura do arquivo com os dados dos empréstimos da

biblioteca, tratou e converteu as datas, criou novas varáveis e agrupou os dados por categorias, NumPy, que serviu para o cálculo das arrays e outras operações numéricas, Matplotlib, que auxiliou na criação e exibição dos gráficos de maneira personalizada com títulos, eixos e legendas, Seaborn, que é baseada na Matplotlib e foi utilizada também para criar gráficos como o de dispersão e desenhar a regressão linear, Scikit-learn, com linear model, que criou e treinou o modelo de regressão linear e gerou os valores previstos com base no modelo ajustado, metrics, que importou as funções para cálculo das métricas do R^2, erro médio, erro quadrático médio e raiz do erro quadrático, e OpenPyXL, utilizado para importar os dados do banco que vieram em uma planilha Excel.

3 METODOLOGIA

O banco de dados utilizado foi o banco da Universidade Federal de São Paulo (Unifesp), na versão de junho de 2023, que contém a relação das obras que no mês foram emprestadas no acervo das bibliotecas da universidade, em Diadema, Guarulhos, Osasco, Santos, São José dos Campos e São Paulo. São apresentadas uma lista das obras e informações a respeito dos empréstimos delas e tem como fonte os relatórios emitidos para a CRBU – Coordenadoria da Rede de Bibliotecas da Unifesp.

O tratamento de dados foi feito a partir da escolha mais recente do banco de dados, depois baixado como planilha do Excel (.xls) e convertido para outra planilha (.xlsx), dessa vez compatível com o leitor Excel atual do Python, todas as categorias de dados foram listadas e separadas em dados categóricos e numéricos e então a análise foi iniciada.

Para o desenvolvimento desse projeto, as ferramentas utilizadas foram o JupyterLab, ambiente de desenvolvimento para notebooks em Python para o desenvolvimento do código, ipykernel, o componente do JupyterLab que executa o código, o jinja2, motor de templates do Jupyter. As bibliotecas Python foram Pandas Python, que analisa dados tabulares, NumPy, que suporta os cálculos feitos durante os processos, Matplotlib e Seaborn, que auxiliaram na criação dos gráficos, Scikit-learn, que auxiliou na construção do modelo de regressão linear, e OpenPyXL, que permitiu a leitura dos arquivos Excel retirados do banco de dados.

Primeiro foram aplicadas as medidas de tendência central e dispersão, seguidas dos testes de normalidade, regressão linear simples, séries temporais e análise de dados categóricos.

4 ANÁLISE DE DADOS

Os métodos de análise previamente citados foram aplicados durante o desenvolvimento desse projeto utilizando as ferramentas descritas no tópico acima com foco em aplicar os conhecimentos adquiridos durante as aulas de Ciência de Dados em um cenário real que banco de dados da biblioteca da UNIFESP proporcionou.

4.1 MEDIDAS DE TENDÊNCIA CENTRAL

As medidas de tendência central foram utilizadas para analisar a relação entre a quantidade de grupos de empréstimo que tiveram a mesma duração, sendo esses grupos, divididos conforme a duração em dias do empréstimo, logo, foram agrupadas juntas, todas as pessoas que mantiveram empréstimos por uma mesma quantidade de dias.

4.1.1 Código aplicado

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = "emprestimos_biblioteca_junho_2023.xlsx"
df = pd.read_excel(data, engine='openpyxl')

df['data_emprestimo'] = pd.to_datetime(df['DATA_EMPRESTIMO_CHAR'], errors='coerce', dayfirst=True)
df['data_devolucao'] = pd.to_datetime(df['DATA_DEV_EFETIVA_CHAR'], errors='coerce', dayfirst=True)

df['dias_emprestados'] = (df['data_devolucao'] - df['data_emprestimo']).dt.days

df = df[df['dias_emprestados'].notnull() & (df['dias_emprestados'] >= 0)]
```

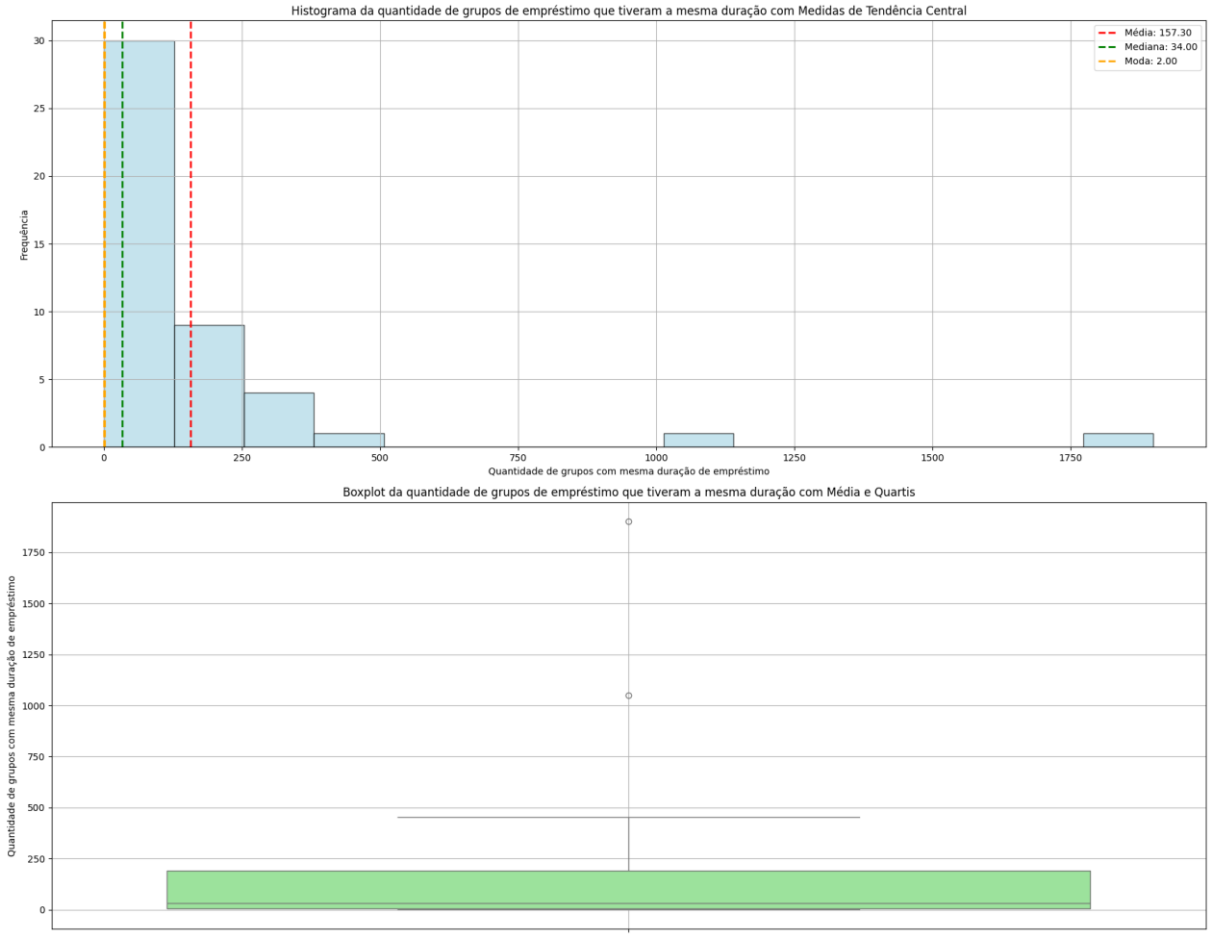
```
df_grouped = df.groupby(['dias_emprestados']).size().reset_index(name='qtd_emprestimos')

# Média
soma = sum(df_grouped['qtd_emprestimos'].values)
quantidade = len(df_grouped['qtd_emprestimos'].values)
media = soma / quantidade

print(f"Média: {media:.2f}")

# Mediana (Usando pandas)
mediana = df_grouped['qtd_emprestimos'].median()
print(f"Mediana: {mediana:.2f}")

# Moda
emprestimos = df_grouped['qtd_emprestimos'].dropna()

def moda(emprestimos):
  frequencia = {}
  for emprestimo in emprestimos:
    frequencia[emprestimo] = frequencia.get(emprestimo, 0) + 1

  frequencia_max = max(frequencia.values())
  modas = [i for i, freq in frequencia.items() if freq == frequencia_max]
  return modas

modas = moda(emprestimos)

print("Moda:", ", ".join(f"{m:.2f}" for m in modas))

data = df_grouped['qtd_emprestimos'].values

mean = media
median = mediana
```

```
modes = moda(emprestimos)

q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
p90 = np.percentile(data, 90)

print(f"Média: {mean:.2f}")
print(f"Mediana: {median:.2f}")
print("Moda(s):", ", ".join(f"{m:.2f}" for m in modes))
print(f"1º Quartil (Q1): {q1:.2f}")
print(f"3º Quartil (Q3): {q3:.2f}")
print(f"Percentil 90 (P90): {p90:.2f}")

fig, axes = plt.subplots(2, 1, figsize=(18, 14))

# Histograma com as linhas da Média e Mediana
axes[0].hist(data, bins=15, color='lightblue', edgecolor='black', alpha=0.7)
axes[0].axvline(mean, color='red', linestyle='dashed', linewidth=2, label=f'Média: {mean:.2f}')
axes[0].axvline(median, color='green', linestyle='dashed', linewidth=2, label=f'Mediana: {median:.2f}')
for i, m in enumerate(modes):
    label = f'Moda: {m:.2f}' if i == 0 else None  # Apenas a primeira linha terá legenda
    axes[0].axvline(m, color='orange', linestyle='dashed', linewidth=2, label=label)
axes[0].legend()
axes[0].set_title('Histograma da quantidade de grupos de empréstimo que tiveram a mesma duração com Medidas de Tendência Central')
axes[0].set_xlabel('Quantidade de grupos com mesma duração de empréstimo')
axes[0].set_ylabel('Frequência')
axes[0].grid(True)
```

```
# Boxplot para visualizar os Quartis e a Mediana
sns.boxplot(data=data, color='lightgreen', ax=axes[1])
axes[1].set_title('Boxplot da quantidade de grupos de empréstimo que tiveram a mesma duração com Média e Quartis')
axes[1].set_ylabel('Quantidade de grupos com mesma duração de empréstimo')
axes[1].grid(True)

plt.tight_layout()
plt.show()

summary = pd.DataFrame({
    'Média': [mean],
    'Mediana': [median],
    'Moda': [", ".join(f"{m:.2f}" for m in modes)],
    '1º Quartil (Q1)': [q1],
    '3º Quartil (Q3)': [q3],
    'Percentil 90 (P90)': [p90]
})

summary
```

4.1.2 Retorno e visualização

Figura 1 - Gráficos de Medidas de Tendência Central



Fonte: própria (2025).

Figura 2 - Resultados valorados

	Média	Mediana	Moda	1º Quartil (Q1)	3º Quartil (Q3)	Percentil 90 (P90)
0	157.304348	34.0	2.00, 1.00	7.0	191.25	357.0

Fonte: própria (2025).

4.1.3 Resultado

A partir dos resultados obtidos, pode-se concluir que existem em média 157 grupos de quantidade de dias que os empréstimos são mantidos, a mediana, moda e média tem valores muito diferentes, o que indica uma irregularidade no registro desses

dados, como comprovado pelos quartis, que apresentam uma irregularidade significativa entre os registros no 1º e 3 º quartil.

4.2 MEDIDAS DE DISPERSÃO

As medidas de dispersão foram utilizadas para analisar a relação entre os grupos de empréstimo que tiveram a mesma duração, nesse caso, foi possível observar que costumam existir mais empréstimos que tem de duração de até 15 dias, mas houve registros com mais de 40 dias, além disso, o desvio padrão está distante da média o que indica dispersão, como confirmado pela amplitude, que está alta.

As diferenças no segundo caso analisado, que verifica a quantidade de dias em que os empréstimos estiveram ativos, são mais brandas, com valores mais próximos, mostrando maior regularidade, entretanto, com diferenças ainda significativas.

4.2.1 Código aplicado

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data = "emprestimos_biblioteca_junho_2023.xlsx"
df = pd.read_excel(data, engine='openpyxl')

df['data_emprestimo'] = pd.to_datetime(df['DATA_EMPRESTIMO_CHAR'], errors='coerce', dayfirst=True)
df['data_devolucao'] = pd.to_datetime(df['DATA_DEV_EFETIVA_CHAR'], errors='coerce', dayfirst=True)

df['dias_emprestados'] = (df['data_devolucao'] - df['data_emprestimo']).dt.days

df = df[df['dias_emprestados'].notnull() & (df['dias_emprestados'] >= 0)]

df_grouped = df.groupby(['dias_emprestados']).size().reset_index(name='qtd_emprestimos')
```

```
# Amplitude
def amplitude(emprestimos):
    return emprestimos.max() - emprestimos.min()

print(f"Números (Quantidade de grupos de empréstimo que tiveram a mesma duração):\n{df_grouped['qtd_emprestimos'].values}")
print(f"Amplitude (Quantidade de grupos de empréstimo que tiveram a mesma duração): {amplitude(df_grouped['qtd_emprestimos'])}")
print(f"Amplitude (Quantidade de dias emprestados): {amplitude(df_grouped['dias_emprestados'])}")

# Variância
def variancia(emprestimos):
  media = sum(emprestimos) / len(emprestimos)
  return sum((i - media) ** 2 for i in emprestimos) / len(emprestimos)

print(f"Números (Quantidade de grupos de empréstimo que tiveram a mesma duração): {df_grouped['qtd_emprestimos'].values}")
print(f"Variância (Quantidade de grupos de empréstimo que tiveram a mesma duração): {variancia(df_grouped['qtd_emprestimos'])}")
print(f"Variância (Quantidade de dias emprestados): {variancia(df_grouped['dias_emprestados'])}")


# Desvio Padrão
def desvio_padrao(emprestimos):
  return variancia(emprestimos) ** 0.5

print(f"Números (Quantidade de grupos de empréstimo que tiveram a mesma duração): {df_grouped['qtd_emprestimos'].values}")
print(f"Desvio Padrão (Quantidade de grupos de empréstimo que tiveram a mesma duração): {desvio_padrao(df_grouped['qtd_emprestimos'])}")
```

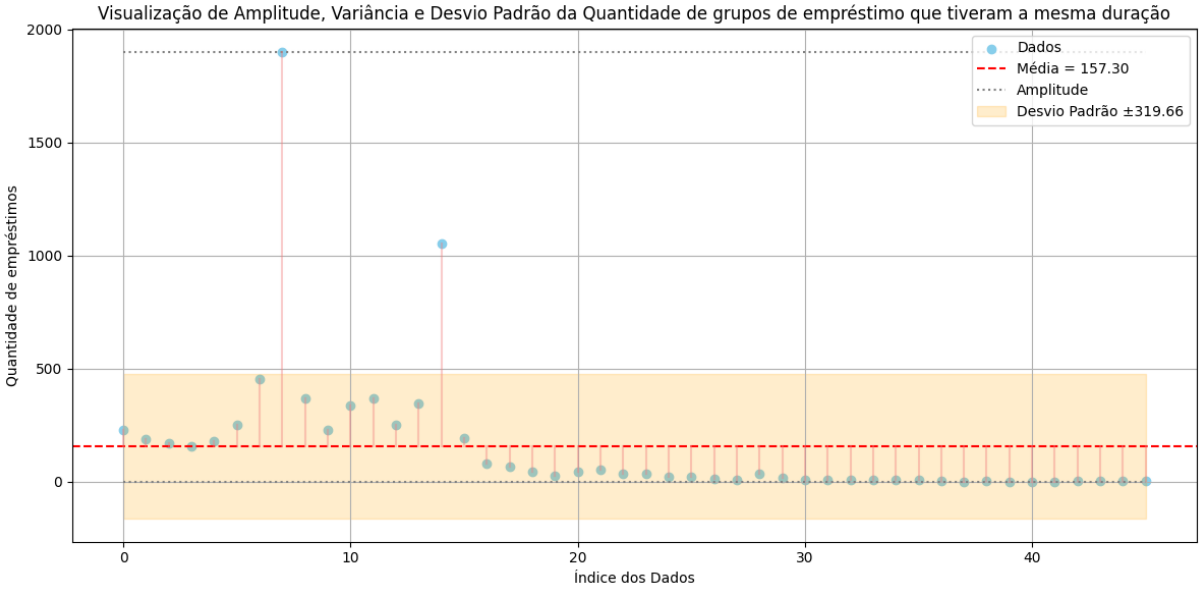
```
print(f"Desvio Padrão (Quantidade de dias emprestados): {desvio_padrao(df_grouped['dias_emprestados'])}")


media = np.mean(df_grouped['qtd_emprestimos'].values)
minimo = np.min(df_grouped['qtd_emprestimos'])
maximo = np.max(df_grouped['qtd_emprestimos'])

plt.figure(figsize=(12, 6))

plt.scatter(range(len(df_grouped['qtd_emprestimos'])), df_grouped['qtd_emprestimos'], color='skyblue', label='Dados')

plt.axhline(media, color='red', linestyle='--', label=f'Média = {media:.2f}')

plt.hlines(y=[minimo, maximo], xmin=0, xmax=len(df_grouped['qtd_emprestimos'].values)-1, colors='gray', linestyles='dotted', label='Amplitude')

desvio = desvio_padrao(df_grouped['qtd_emprestimos'].values)

plt.fill_between(range(len(df_grouped['qtd_emprestimos'])), media - desvio, media + desvio,
                 color='orange', alpha=0.2, label=f'Desvio Padrão ±{desvio:.2f}')

for i, y in enumerate(df_grouped['qtd_emprestimos']):
    plt.plot([i, i], [media, y], color='lightcoral', alpha=0.4)

plt.title("Visualização de Amplitude, Variância e Desvio Padrão da Quantidade de grupos de empréstimo que tiveram a mesma duração")
plt.xlabel("Índice dos Dados")
plt.ylabel("Quantidade de empréstimos")
plt.legend()
plt.grid(True)
```

```
plt.tight_layout()
plt.show()

tabela = pd.DataFrame({
    'Medida': ['Média', 'Desvio Padrão', 'Variância', 'Quantidade Mínima', 'Quantidade Máxima', 'Amplitude'],
    'Quantidade': [
        media,
        desvio_padrao(df_grouped['qtd_emprestimos']),
        variancia(df_grouped['qtd_emprestimos']),
        minimo,
        maximo,
        amplitude(df_grouped['qtd_emprestimos'])
    ]
})

tabela.style.format({'Quantidade': '{:.2f}'})


media = np.mean(df_grouped['dias_emprestados'].values)
minimo = np.min(df_grouped['dias_emprestados'])
maximo = np.max(df_grouped['dias_emprestados'])

plt.figure(figsize=(12, 6))

plt.scatter(range(len(df_grouped['dias_emprestados'])), df_grouped['dias_emprestados'], color='skyblue', label='Dados')

plt.axhline(media, color='red', linestyle='--', label=f'Média = {media:.2f}')

plt.hlines(y=[minimo, maximo], xmin=0, xmax=len(df_grouped['dias_emprestados'].values)-1, colors='gray', linestyles='dotted', label='Amplitude')
```

```
desvio = desvio_padrao(df_grouped['dias_emprestados'].values)

plt.fill_between(range(len(df_grouped['dias_emprestados'])), media - desvio, media + desvio,
                 color='orange', alpha=0.2, label=f'Desvio Padrão ±{desvio:.2f}')

for i, y in enumerate(df_grouped['dias_emprestados']):
    plt.plot([i, i], [media, y], color='lightcoral', alpha=0.4)

plt.title("Visualização de Amplitude, Variância e Desvio Padrão da Quantidade de dias emprestados")
plt.xlabel("Índice dos Dados")
plt.ylabel("Quantidade")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

tabela = pd.DataFrame({
    'Medida': ['Média', 'Desvio Padrão', 'Variância', 'Quantidade Mínima', 'Quantidade Máxima', 'Amplitude'],
    'Quantidade': [
        media,
        desvio_padrao(df_grouped['dias_emprestados']),
        variancia(df_grouped['dias_emprestados']),
        minimo,
        maximo,
        amplitude(df_grouped['dias_emprestados'])
    ]
})

tabela.style.format({'Quantidade': '{:.2f}'})
```

4.2.2 Retorno e visualização

Figura 3 - Gráfico sobre grupos com mesma duração



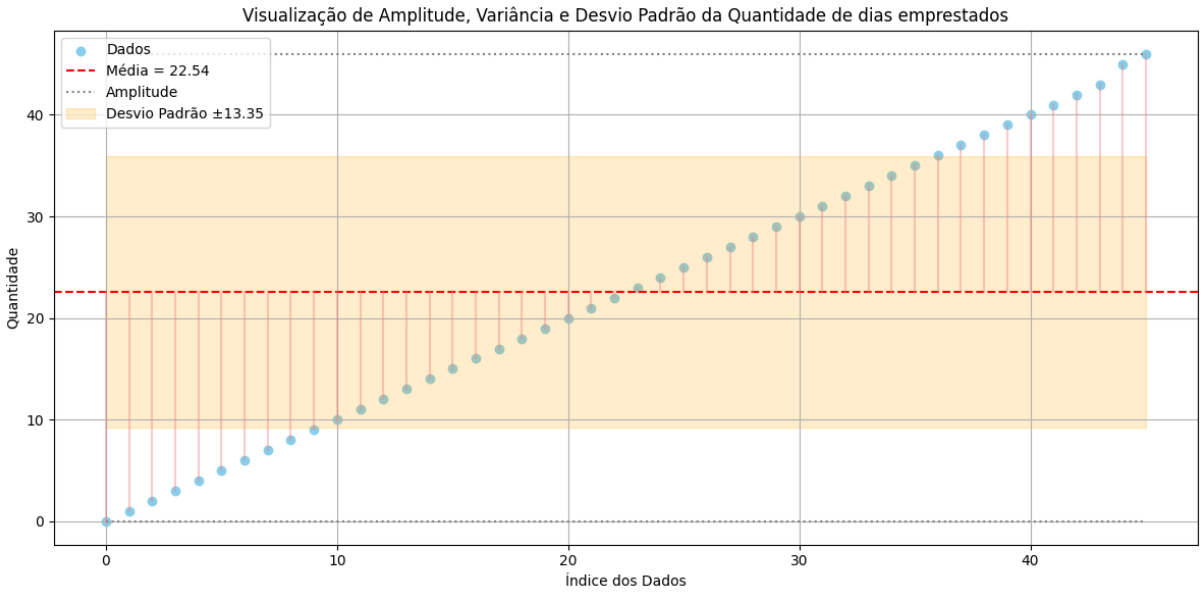
Fonte: própria (2025).

Figura 4 - Resultados numéricos

	Medida	Valor
0	Média	157.30
1	Desvio Padrão	319.66
2	Variância	102183.78
3	Valor Mínimo	1.00
4	Valor Máximo	1900.00
5	Amplitude	1899.00

Fonte: própria (2025).

Figura 5 - Gráfico sobre a quantidade de dias



Fonte: própria (2025).

Figura 6 - Resultados numéricos

	Medida	Valor
0	Média	22.54
1	Desvio Padrão	13.35
2	Variância	178.20
3	Valor Mínimo	0.00
4	Valor Máximo	46.00
5	Amplitude	46.00

Fonte: própria (2025).

4.2.3 Resultado

Foi possível concluir que os perfis de empréstimo apresentam grande variação e a quantidade para certas datas terem mais registros tem potencial para investigação causal junto a biblioteca de origem.

4.3 TESTES DE NORMALIDADE

Os testes de normalidade foram aplicados para analisar a relação entre os grupos de empréstimo que tiveram a mesma duração.

4.3.1 Código aplicado

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency

data = "emprestimos_biblioteca_junho_2023.xlsx"
df = pd.read_excel(data, engine='openpyxl')

# Chi
tabela = pd.crosstab(df['DESC_CATEG_USUARIO'], df['Nome tipo obra'])

chi2, p, dof, expected = chi2_contingency(tabela)

# Diferença entre observado e esperado
expected = pd.DataFrame(expected, index=tabela.index, columns=tabela.columns)
diff = tabela - expected

fig, axes = plt.subplots(2, 1, figsize=(20, 14))

# Gráfico 1: barras empilhadas
cores = ['#a6cee3', '#1f78b4', '#b2df8a', "#ff00ea", "#e9ff25"]
tabela.plot(kind='bar', stacked=True, color=cores, ax=axes[0], edgecolor='black')
axes[0].set_title('Distribuição por Tipo de Usuário e Tipo de Obra emprestada', fontsize=13)
axes[0].set_xlabel('Tipo de Usuário')
axes[0].set_ylabel('Quantidade de empréstimos')
axes[0].legend(title='Tipo de Obra', bbox_to_anchor=(1.05, 1), loc='upper left')
axes[0].grid(axis='y', linestyle='--', alpha=0.7)
axes[0].tick_params(axis='x', rotation=90)
```

```
        # Gráfico 2: mapa de calor da diferença (observado - esperado)
        sns.heatmap(diff, annot=True, fmt=".1f", cmap="coolwarm", center=0, ax=axes[1], annot_kws={"size": 7})
        axes[1].set_title('Desvio entre Observado e Esperado')
        axes[1].set_xlabel('Tipo de Obra')
        axes[1].set_ylabel('Tipo de Usuário')

        plt.tight_layout()
        plt.subplots_adjust(right=0.87)
        plt.show()

        df['data_emprestimo'] = pd.to_datetime(df['DATA_EMPRESTIMO_CHAR'], errors='coerce', dayfirst=True)
        df['data_devolucao'] = pd.to_datetime(df['DATA_DEV_EFETIVA_CHAR'], errors='coerce', dayfirst=True)

        df['dias_emprestados'] = (df['data_devolucao'] - df['data_emprestimo']).dt.days

        df = df[df['dias_emprestados'].notnull() & (df['dias_emprestados'] >= 0)]

        df_grouped = df.groupby(['dias_emprestados']).size().reset_index(name='qtd_emprestimos')

        # Shapiro-Wilk
        from scipy.stats import shapiro

        stat, p = shapiro(df_grouped['qtd_emprestimos'].values)

        print("Estatística W:", round(stat, 4))
        print("p-valor:", round(p, 4))
        if p < 0.05:
            print("Resultado: Os dados NÃO seguem uma distribuição normal (rejeita H₀)")
```

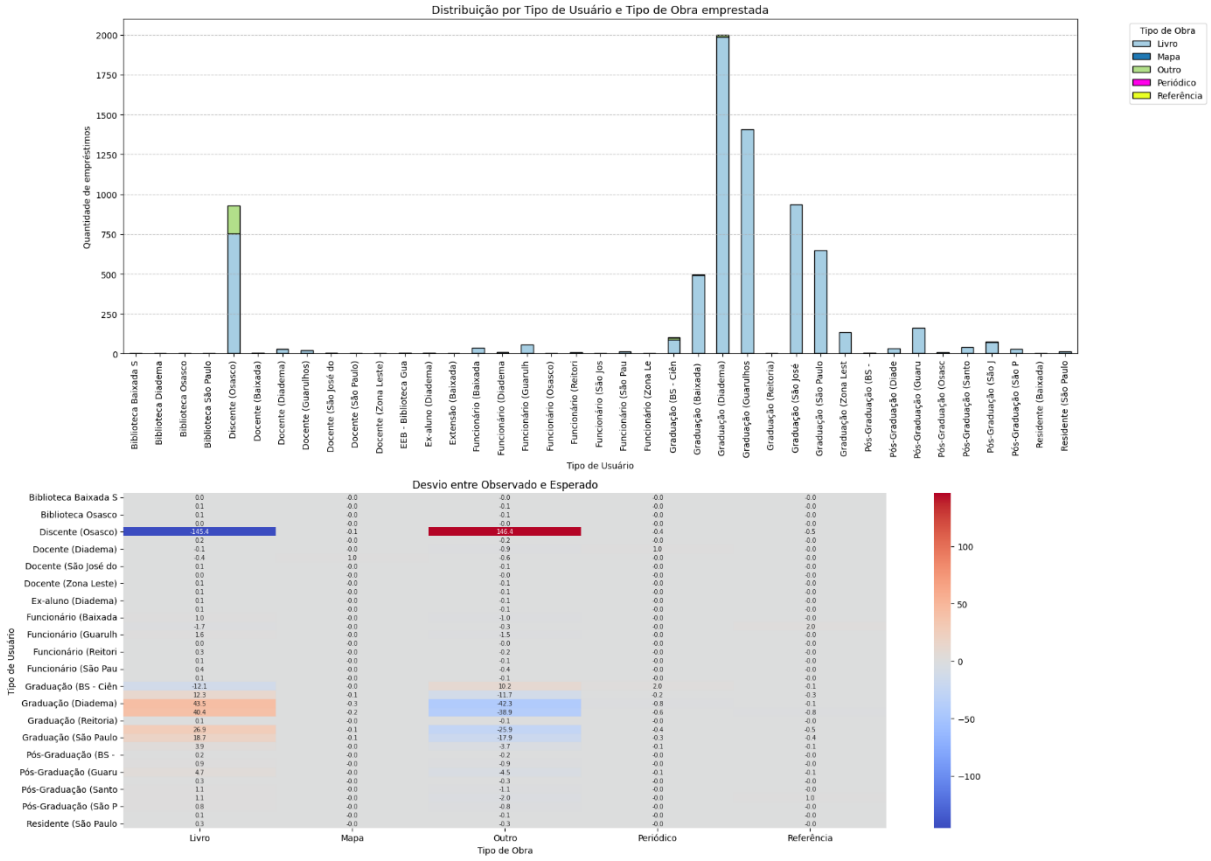
```
else:
    print("Resultado: Os dados seguem uma distribuição normal (não rejeita H₀)")

plt.figure(figsize=(10, 5))
sns.histplot(df_grouped['qtd_emprestimos'].values, kde=True, bins=10, color='skyblue')
plt.title("Distribuição dos Dados - Shapiro-Wilk - De grupos de empréstimo que tiveram a mesma duração")
plt.xlabel("Quantidade de Empréstimos")
plt.ylabel("Frequência")
plt.grid(True)
plt.tight_layout()
plt.show()

# Teste Kolmogorov-Smirnov (K-S)
from scipy.stats import kstest, norm

dados_norm = (df_grouped['qtd_emprestimos'].values - np.mean(df_grouped['qtd_emprestimos'].values)) / np.std(df_grouped['qtd_emprestimos'].values)

stat, p = kstest(dados_norm, 'norm')

print("Estatística D:", round(stat, 4))
print("p-valor:", round(p, 4))
if p < 0.05:
    print("Resultado: Os dados NÃO seguem a distribuição normal (rejeta H₀)")
else:
    print("Resultado: Os dados seguem a distribuição normal (não rejeita H₀)")

plt.figure(figsize=(10, 5))
sns.histplot(dados_norm, kde=True, color='mediumseagreen', bins=20)
```

```
plt.title("Distribuição dos Dados Normalizados - Teste K-S - De grupos de empréstimo que tiveram a mesma duração")
plt.xlabel("Z-score")
plt.ylabel("Frequência")
plt.grid(True)
plt.tight_layout()
plt.show()

# Teste Anderson-Darling
from scipy.stats import anderson

resultado = anderson(df_grouped['qtd_emprestimos'].values, dist='norm')

print("Estatística A²:", round(resultado.statistic, 4))
print("\nValores críticos e significância:")
for critico, sig in zip(resultado.critical_values, resultado.significance_level):
    print(f"  {sig}% : {critico:.4f}")

# Veredito com base no nível de 5%
nivel_significancia = 5
indice = resultado.significance_level.tolist().index(nivel_significancia)
if resultado.statistic > resultado.critical_values[indice]:
    print("\nResultado: Os dados NÃO seguem uma distribuição normal (rejeita H₀)")
else:
    print("\nResultado: Os dados seguem uma distribuição normal (não rejeita H₀)")

plt.figure(figsize=(10, 5))
sns.histplot(df_grouped['qtd_emprestimos'].values, kde=True, color='slateblue', bins=20)
plt.title("Distribuição dos Dados - Teste Anderson-Darling - De grupos de empréstimo que tiveram a mesma duração")
plt.xlabel("Quantidade de Empréstimos")
```

```
plt.ylabel("Frequência")
plt.grid(True)
plt.tight_layout()
plt.show()
```

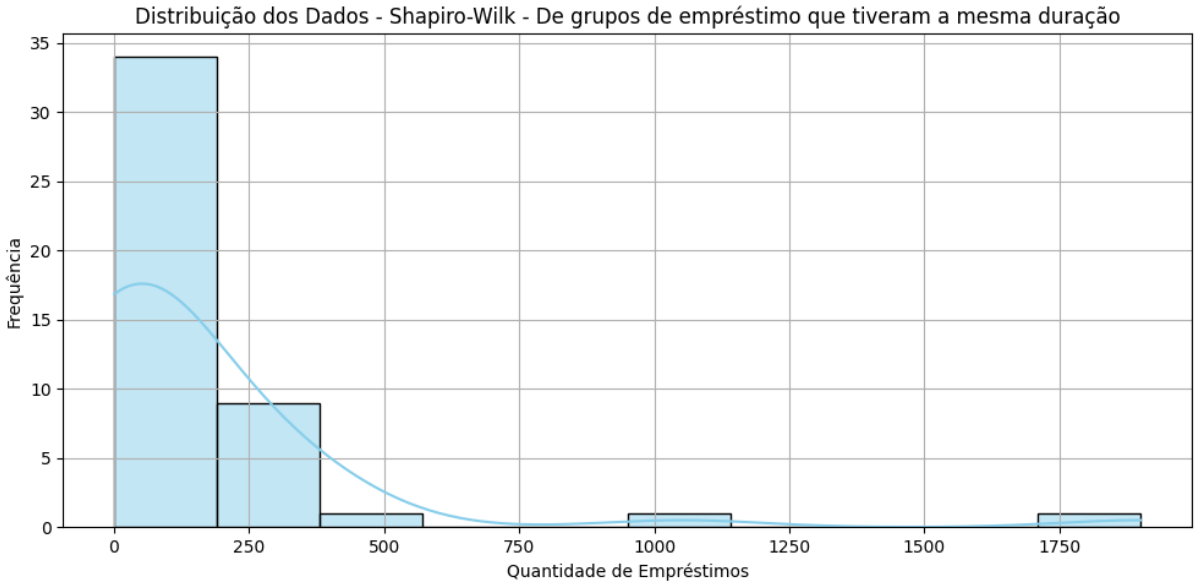
4.3.2 Retorno e visualização

Figura 7 - Gráficos de distribuição por tipo de usuário e tipo de obra emprestada, e desvio entre observado e esperado respectivamente



Fonte: própria (2025).

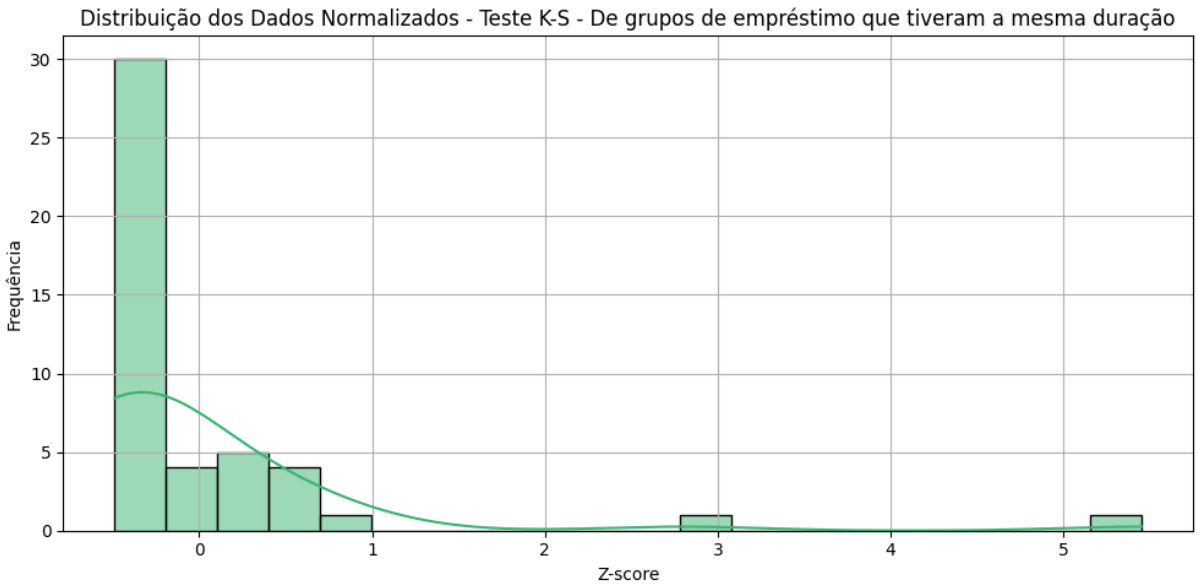
Figura 8 - Shapiro-Wilk



Fonte: própria (2025).

Resultados numéricos de Shapiro-Wilk: Estatística W como 0.5011 e p-valor como 0.0.

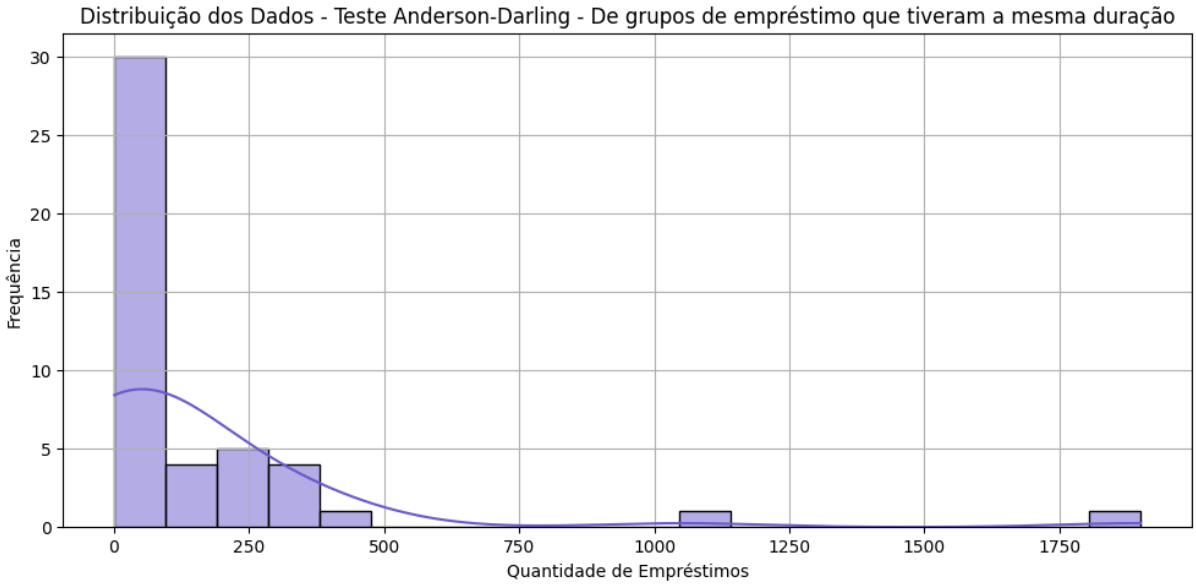
Figura 9 - Kolmogorov-Smirnov



Fonte: própria (2025).

Resultados numéricos de Kolmogorov-Smirnov: Estatística D como 0.3124 e p-valor como 0.0002.

Figura 10 - Anderson-Darling



Fonte: própria (2025).

Resultados numéricos de Anderson-Darling: Estatística A^2 como 7.0355 e valores críticos e de significância 15%, 10%, 5%, 2.5% e 1% respectivamente 0.5306, 0.6100, 0.7320. 0.8540 e 1.0160.

4.3.3 Resultado

Como é possível observar nos gráficos, os dados analisados não seguem uma distribuição normal.

4.4 REGRESSÃO LINEAR SIMPLES

A regressão linear simples foi utilizada para analisar o tempo de empréstimo em contraste com a quantidade de obras emprestadas.

4.4.1 Código aplicado

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
import numpy as np
```

```
data = "emprestimos_biblioteca_junho_2023.xlsx"
df = pd.read_excel(data, engine='openpyxl')

df['data_emprestimo'] = pd.to_datetime(df['DATA_EMPRESTIMO_CHAR'], errors='coerce', dayfirst=True)
df['data_devolucao'] = pd.to_datetime(df['DATA_DEV_EFETIVA_CHAR'], errors='coerce', dayfirst=True)

df['dias_emprestados'] = (df['data_devolucao'] - df['data_emprestimo']).dt.days

df = df[df['dias_emprestados'].notnull() & (df['dias_emprestados'] >= 0)]

df_grouped = df.groupby(['dias_emprestados']).size().reset_index(name='qtd_emprestimos')

# Treinar o modelo e prever
X = df_grouped[['dias_emprestados']]
y = df_grouped['qtd_emprestimos']

modelo = LinearRegression()
modelo.fit(X, y)

df_grouped['predict'] = modelo.predict(X)

a = modelo.intercept_
b = modelo.coef_[0]
print(f"Quantidade de obras emprestadas = {b:.2f} * Tempo de empréstimo em dias + {a:.2f}")

# Gráfico de dispersão
sns.scatterplot(x="dias_emprestados", y="qtd_emprestimos", data=df_grouped)
```

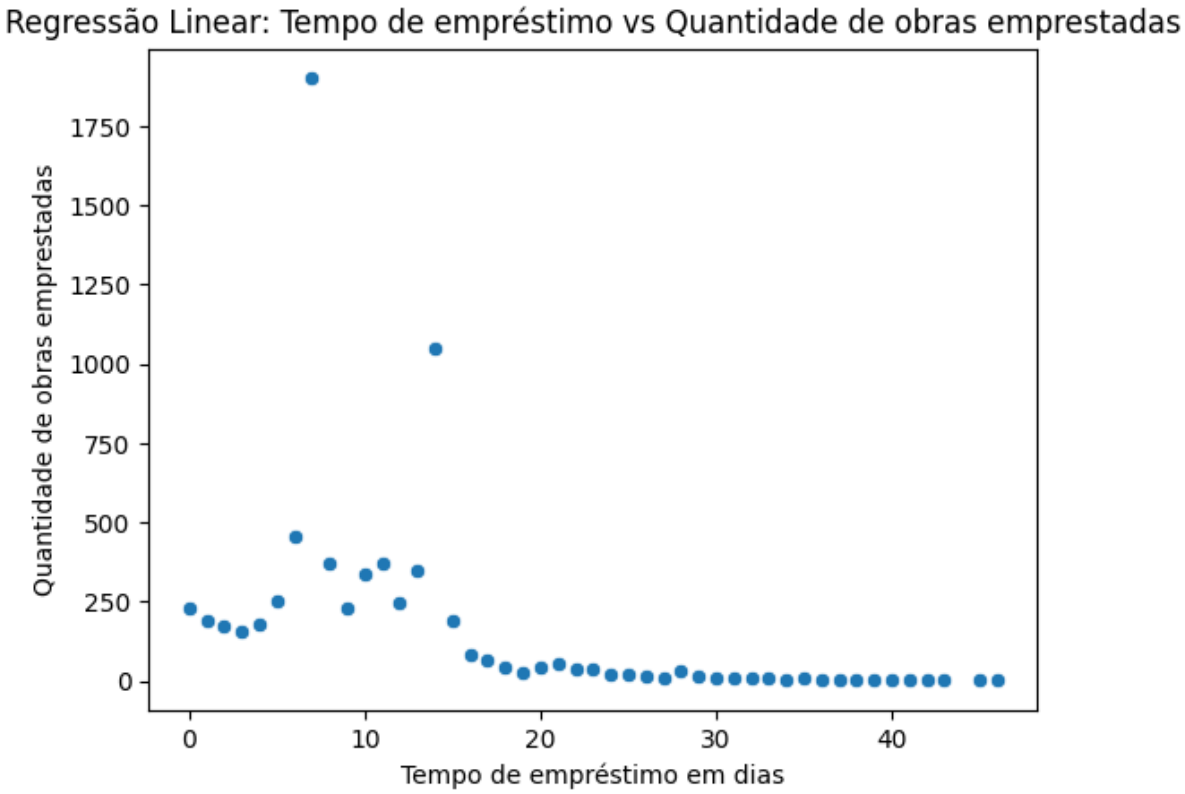
```
plt.title("Regressão Linear: Tempo de empréstimo vs Quantidade de obras emprestadas")
plt.xlabel("Tempo de empréstimo em dias")
plt.ylabel("Quantidade de obras emprestadas")
plt.show()

# Linha de regressão
plt.figure(figsize=(10, 5))
sns.scatterplot(x='dias_emprestados', y='qtd_emprestimos', data=df_grouped, label='Real')
sns.lineplot(x='dias_emprestados', y='predict', data=df_grouped, color='red', label='Regressão Linear')
plt.title(f'Regressão Linear: Tempo de empréstimo vs Quantidade de obras emprestadas')
plt.xlabel('Tempo de empréstimo em dias')
plt.ylabel('Quantidade de obras emprestadas')
plt.legend()
plt.show()

# Avaliar o modelo
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

r2 = r2_score(y, df_grouped["predict"])
mae = mean_absolute_error(y, df_grouped["predict"])
mse = mean_squared_error(y, df_grouped["predict"])
rmse = mse ** 0.5

print(f'R²: {r2:.2f}')
print(f'MAE: {mae:.2f}')
print(f'MSE: {mse:.2f}')
print(f'RMSE: {rmse:.2f}')

# Análise de resíduos
```

```
df_grouped['residuos'] = df_grouped['qtd_emprestimos'] - df_grouped['predict']

plt.figure(figsize=(10, 5))
sns.scatterplot(x=df_grouped['dias_emprestados'], y=df_grouped['residuos'])
plt.axhline(0, color='red', linestyle='--')
plt.title('Resíduos da Regressão Linear')
plt.xlabel('Tempo de empréstimo em dias')
plt.ylabel('Resíduo (Real - Previsto)')
plt.show()
```

4.4.2 Retorno e visualização

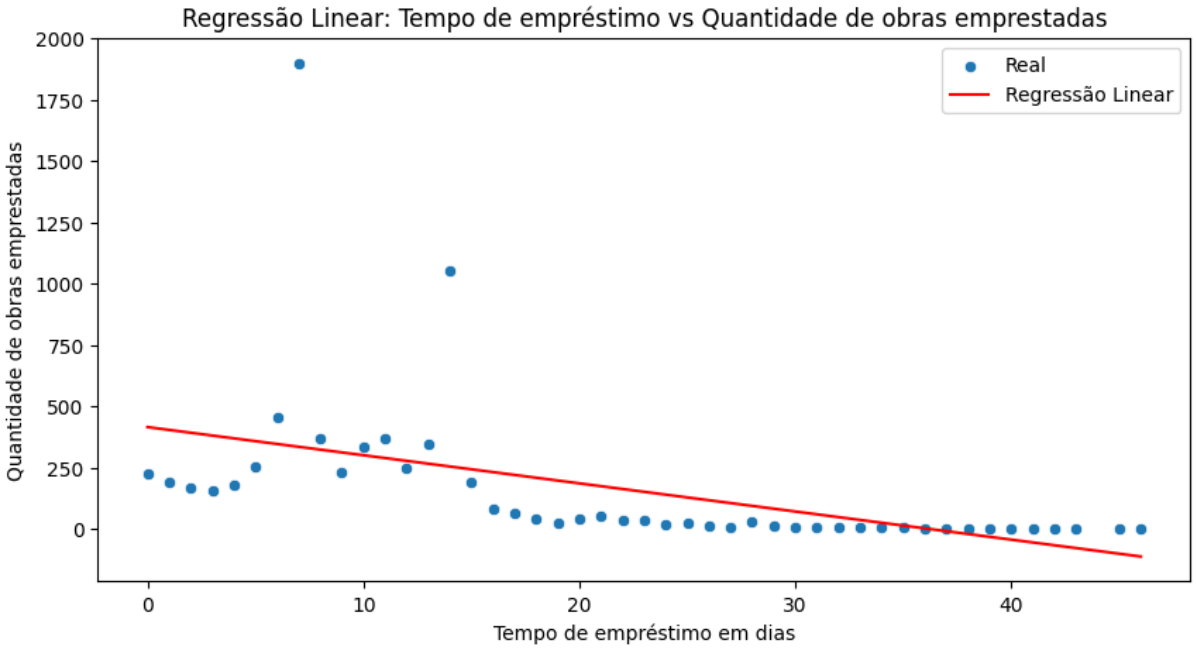
O modelo utilizou a seguinte fórmula para análise: Quantidade de obras emprestadas = -11.47 * Tempo de empréstimo em dias + 415.90.

Figura 11 - Gráfico de dispersão



Fonte: própria (2025).

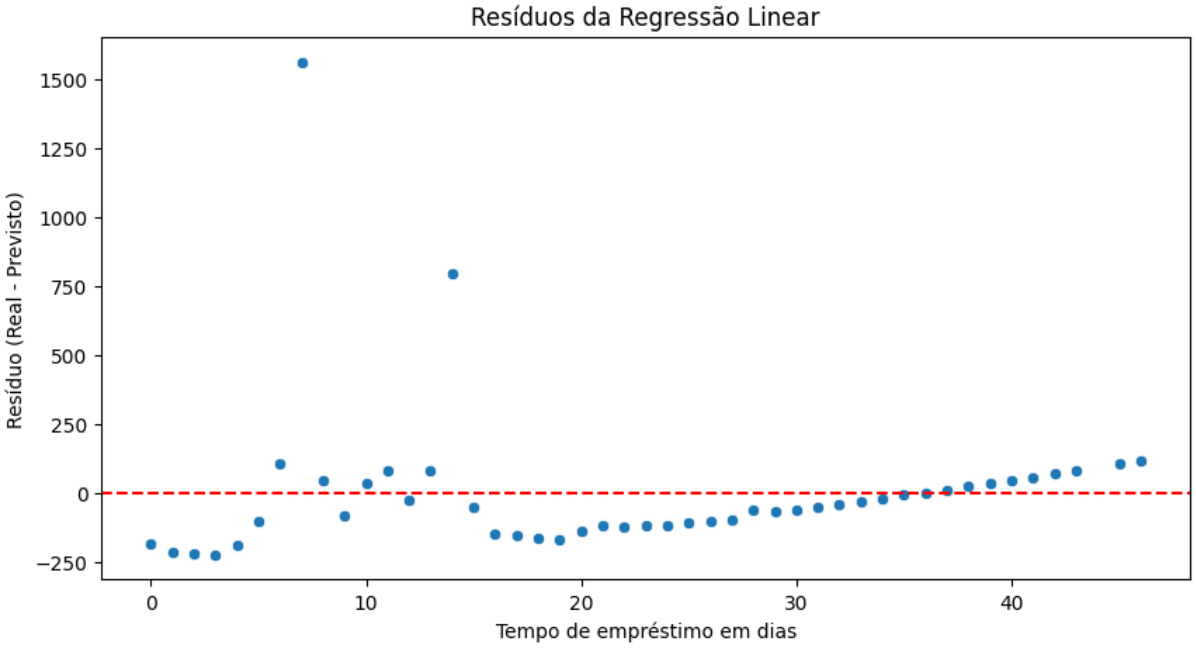
Figura 12 - Linha de regressão



Fonte: própria (2025).

A avaliação do modelo retornou R² como 0.23, MAE como 140.64, MSE como 78735.38 e RMSE como 280.60.

Figura 13 - Análise de resíduos



Fonte: própria (2025).

4.4.3 Resultado

Analisando os gráficos, pode-se concluir que conforme a duração dos empréstimos aumenta, a quantidade de obras emprestadas tende a diminuir, mesmo com algumas exceções.

4.5 SÉRIES TEMPORAIS

As séries temporais foram analisadas com base nos dados dos empréstimos diários.

4.5.1 Código aplicado

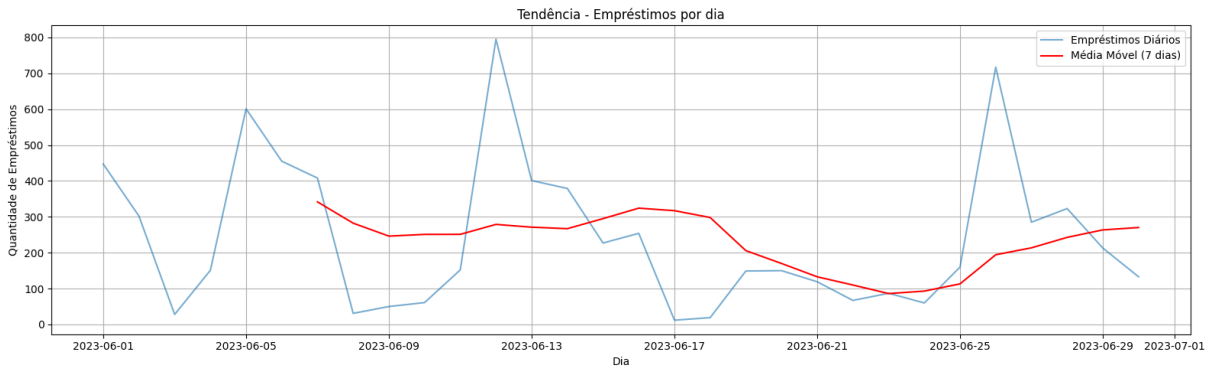
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
import numpy as np

data = "emprestimos_biblioteca_junho_2023.xlsx"
df = pd.read_excel(data, engine='openpyxl')

df['data_emprestimo'] = pd.to_datetime(df['DATA_EMPRESTIMO_CHAR'], errors='coerce', dayfirst=True)
df['data_devolucao'] = pd.to_datetime(df['DATA_DEV_EFETIVA_CHAR'], errors='coerce', dayfirst=True)

df['dias_emprestados'] = (df['data_devolucao'] - df['data_emprestimo']).dt.days

df = df[df['dias_emprestados'].notnull() & (df['dias_emprestados'] >= 0)]

df_grouped = df.groupby(['dias_emprestados']).size().reset_index(name='qtd_emprestimos')

# Tendência
```

```
df['data_emprestimo'] = pd.to_datetime(df['data_emprestimo'])

emprestimos_por_dia = df.groupby('data_emprestimo').size().sort_index()

serie = pd.DataFrame({'qtd_emprestimos': emprestimos_por_dia})
serie['MediaMovel_7'] = serie['qtd_emprestimos'].rolling(window=7).mean()

plt.figure(figsize=(16, 5))
plt.plot(serie['qtd_emprestimos'], label='Empréstimos Diários', alpha=0.6)
plt.plot(serie['MediaMovel_7'], label='Média Móvel (7 dias)', color='red')
plt.title('Tendência - Empréstimos por dia')
plt.xlabel('Dia')
plt.ylabel('Quantidade de Empréstimos')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Ciclo
import locale
locale.setlocale(locale.LC_TIME, 'pt_BR.UTF-8')

df['data_emprestimo'] = pd.to_datetime(df['data_emprestimo'])

df['dia_semana'] = df['data_emprestimo'].dt.day_name(locale='pt_BR')

ordem = ['Segunda-feira', 'Terça-feira', 'Quarta-feira', 'Quinta-feira', 'Sexta-feira',
'Sábado', 'Domingo']

semana = df.groupby('dia_semana').size().reindex(ordem)

plt.figure(figsize=(8, 5))
sns.barplot(x=semana.index, y=semana.values)
plt.title('Ciclo - Empréstimos por dia')
```

```
plt.xlabel('Dia')
plt.ylabel('Quantidade de Empréstimos')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

# Decomposição da Série Temporal
from statsmodels.tsa.seasonal import seasonal_decompose

serie = serie.asfreq('D', fill_value=0)
decomposicao = seasonal_decompose(serie['qtd_emprestimos'], model='additive', period=7)

decomposicao.plot()
plt.suptitle('Decomposição da Série Temporal de Empréstimos', fontsize=16)
plt.tight_layout()
plt.show()
```

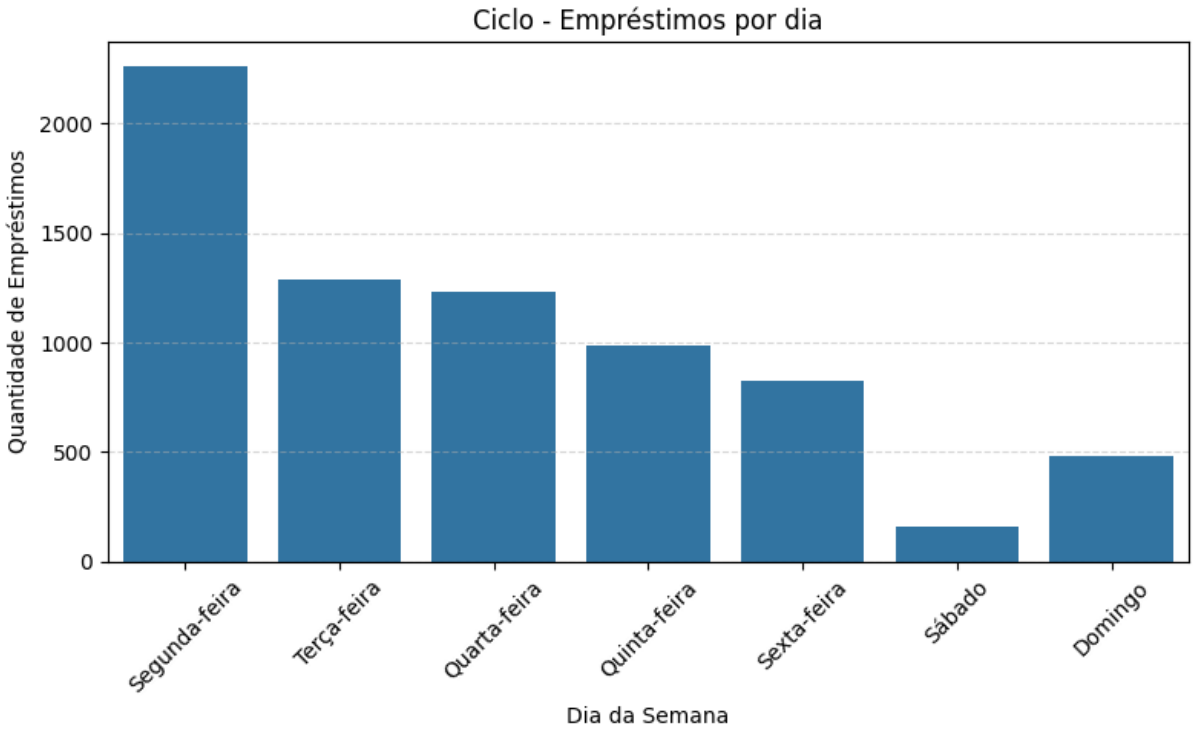
4.5.2 Retorno e visualização

Figura 14 - Tendência



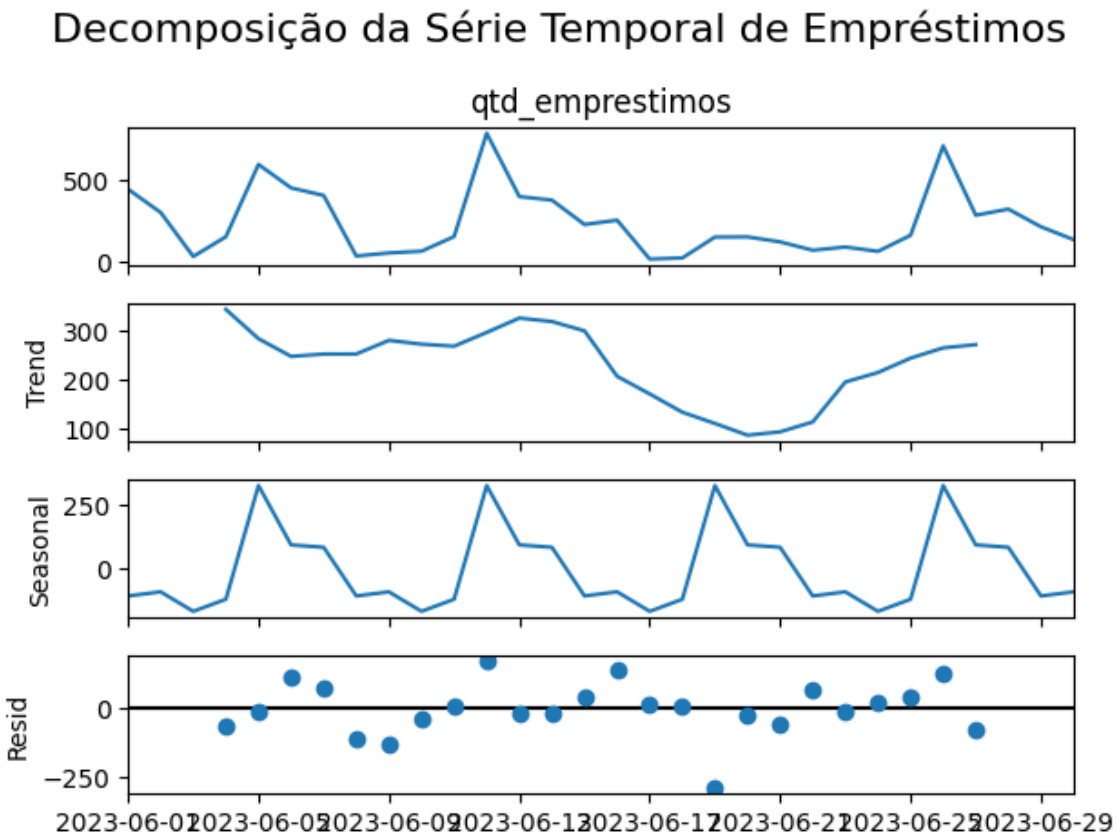
Fonte: própria (2025).

Figura 15 - Ciclo



Fonte: própria (2025).

Figura 16 - Decomposição da Série Temporal



Fonte: própria (2025).

4.5.3 Resultado

De acordo com os gráficos, pode-se concluir que a quantidade de empréstimos tende a cair, mas se recuperando em seguida ao longo das semanas, e que os empréstimos acontecem recorrentemente durante as semanas (como observado na sazonalidade).

4.6 ANÁLISE DE DADOS CATEGÓRICOS

Os dados categóricos analisados foram baseados nos tipos de obras emprestadas conforme os tipos de usuário.

4.6.1 Código aplicado

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data = "emprestimos_biblioteca_junho_2023.xlsx"
df = pd.read_excel(data, engine='openpyxl')

tabela = pd.crosstab(df['DESC_CATEG_USUARIO'], df['Nome tipo obra'])
frequencia = tabela.sum(axis=1)
print(frequencia)

plt.figure(figsize=(16, 6))
frequencia.plot(kind='bar', color='skyblue', title='Quantidade de tipo de obra emprestada conforme tipo de usuário')
plt.xlabel('Tipo de usuário')
plt.ylabel('Quantidade de Obras')
plt.grid(True)
plt.show()

num_categorias = len(frequencia)
```

```
cores_pizza = plt.cm.hsv(np.linspace(0, 1, num_categorias))

plt.figure(figsize=(10, 7))
frequencia.plot.pie(
    colors=cores_pizza,
    labels=[''] * num_categorias,
    legend=False,
    startangle=90,
    title='Porcentagem de empréstimos conforme tipo de usuário'
)
plt.ylabel('')
plt.tight_layout()
plt.legend(frequencia.index, bbox_to_anchor=(1.05, 0.5), loc='center left', fontsize='small')
plt.show()
```

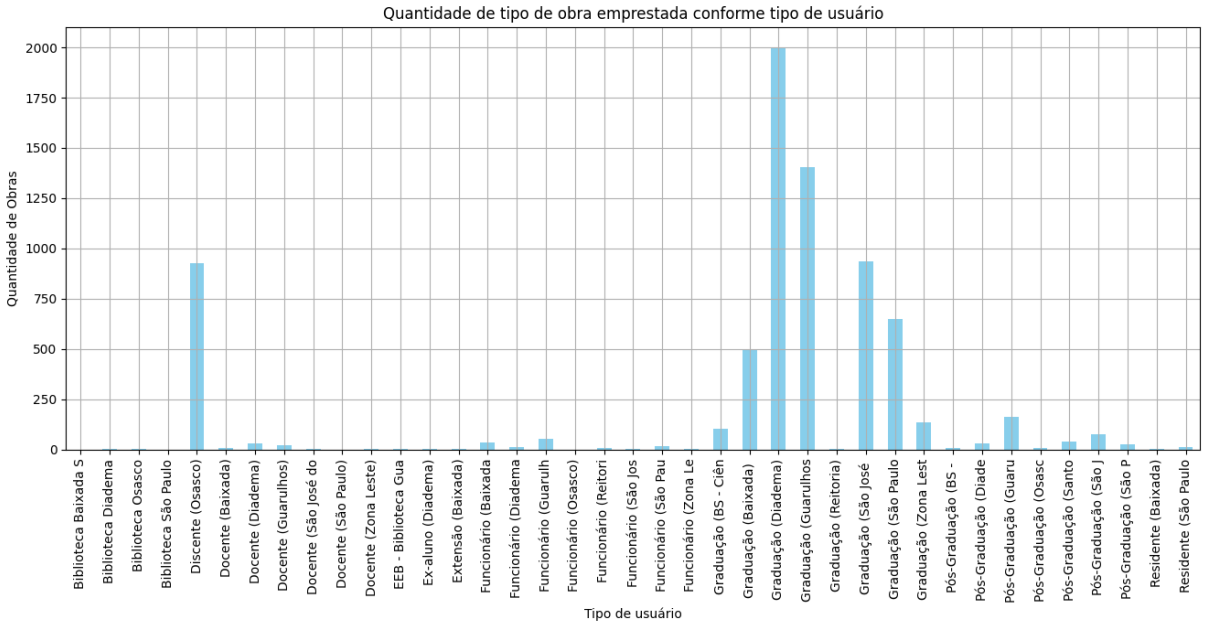
4.6.2 Retorno e visualização

Figura 17 - Quantidade de usuários por categoria

DESC_CATEG_USUARIO	
Biblioteca Baixada S	1
Biblioteca Diadema	2
Biblioteca Osasco	2
Biblioteca São Paulo	1
Discente (Osasco)	927
Docente (Baixada)	6
Docente (Diadema)	31
Docente (Guarulhos)	22
Docente (São José do	5
Docente (São Paulo)	1
Docente (Zona Leste)	3
EEB - Biblioteca Gua	5
Ex-aluno (Diadema)	4
Extensão (Baixada)	2
Funcionário (Baixada	35
Funcionário (Diadema	11
Funcionário (Guarulh	55
Funcionário (Osasco)	1
Funcionário (Reitori	9
Funcionário (São Jos	3
Funcionário (São Pau	15
Funcionário (Zona Le	2
Graduação (BS - Ciên	102
Graduação (Baixada)	496
Graduação (Diadema)	2001
Graduação (Guarulhos	1406
Graduação (Reitoria)	2
Graduação (São José	936
Graduação (São Paulo	649
Graduação (Zona Lest	134
Pós-Graduação (BS -	7
Pós-Graduação (Diade	32
Pós-Graduação (Guaru	163
Pós-Graduação (Osasc	10
Pós-Graduação (Santo	39
Pós-Graduação (São J	74
Pós-Graduação (São P	28
Residente (Baixada)	2
Residente (São Paulo	12

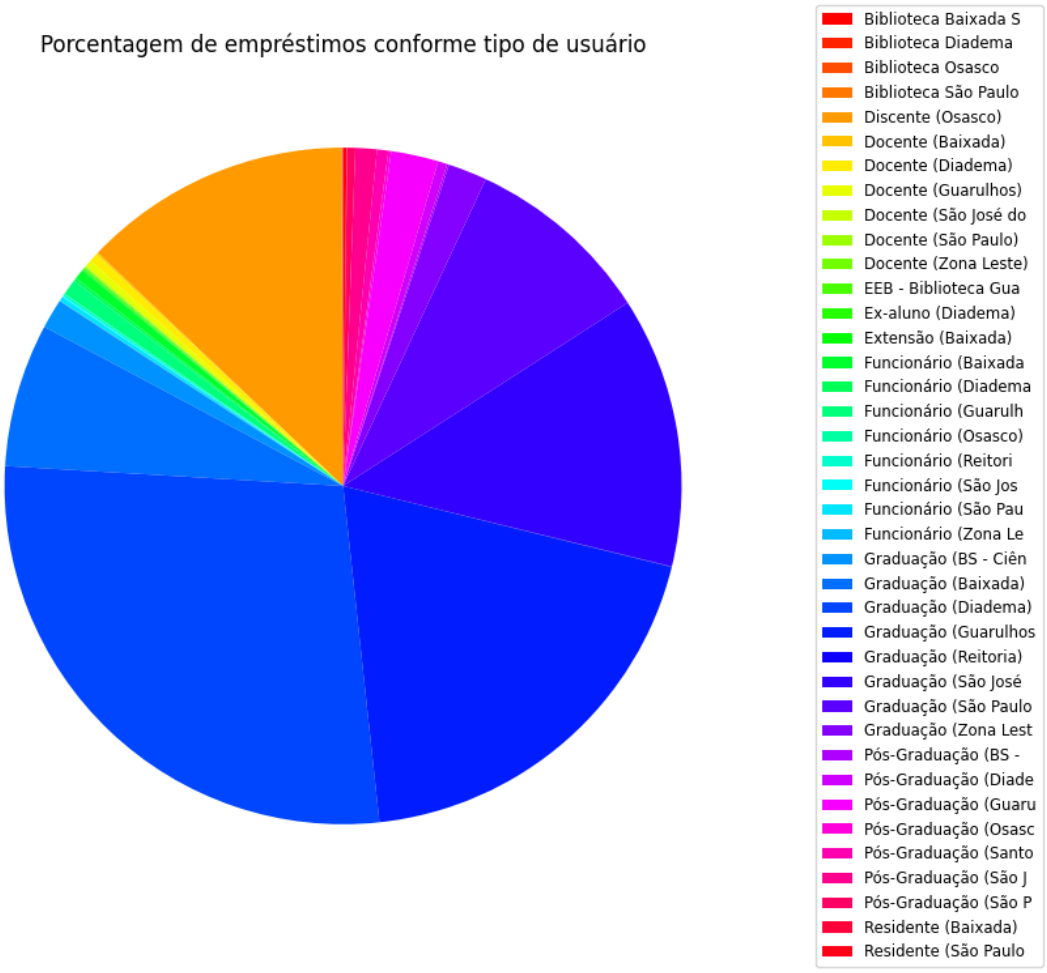
Fonte: própria (2025).

Figura 18 - Quantidade de tipo de obra conforme o tipo de usuário



Fonte: própria (2025).

Figura 19 - Porcentagem de empréstimos conforme tipo de usuário



Fonte: própria (2025).

4.6.3 Resultado

Conforme os gráficos, é possível concluir que os usuários que mais emprestam livros de diversos tipos são os estudantes de graduação de diversos campus, seguidos dos docentes de Osasco.

5 CONCLUSÃO

O desenvolvimento deste projeto permitiu aplicar conceitos fundamentais da Ciência de Dados em um cenário realista, voltado à organização e análise de informações de uma biblioteca. Ao longo do trabalho, foi possível estruturar dados de maneira organizada, simular interações entre diferentes entidades (como usuários, livros e empréstimos), e preparar a base para análises futuras.

Entre os principais achados, destacam-se a importância de manter os dados limpos e organizados para facilitar futuras análises e visualizações, a compreensão da lógica relacional entre os dados e sua aplicação prática em contextos do cotidiano, como controle de empréstimos, e a noção clara da preparação de dados como uma das etapas mais críticas antes de qualquer modelo preditivo.

Apesar dos avanços, algumas limitações foram encontradas como a ausência da aplicação direta de modelos estatísticos ou algoritmos de aprendizado de máquina, o que reduz a complexidade da análise e o tempo para aprofundar a visualização de dados e gerar insights mais elaborados foi limitado.

Como sugestões para trabalhos futuros, destacam-se a ampliação da base de dados com registros reais ou simulados em maior volume, a aplicação de técnicas de visualização (como gráficos interativos) para facilitar o entendimento dos padrões nos dados e a introdução de modelos estatísticos e preditivos para identificar comportamentos, como previsão de devoluções ou perfis de usuários mais ativos.

REFERÊNCIAS

BRASIL. Ministério do Planejamento, Orçamento e Gestão. Empréstimos Biblioteca. Dados Abertos. Disponível em: https://dados.gov.br/dados/conjuntos-dados/emprestimos_biblioteca. Acesso em: 03 jun. 2025.

GEEKSFORGEEKS. How to convert categorical string data into numeric in Python. GeeksforGeeks, 2024. Disponível em: https://www.geeksforgeeks.org/python/how-to-convert-categorical-string-data-into-numeric-in-python/. Acesso em: 03 jun. 2025.

GITHUB. *Material/Ciência de Dados/*. profAndreSouza, c2025. Disponível em: < https://github.com/profAndreSouza/Material/tree/main/Ci%C3%AAncia%20de%20Dados> Acesso em: 03 jun. 2025.

IPYKERNEL. ipykernel. PyPI, 2025. Disponível em: https://pypi.org/project/ipykernel/. Acesso em: 03 jun. 2025.

JINJA2. Jinja2 Documentation. PyPI, 2025. Disponível em: https://pypi.org/project/Jinja2/. Acesso em: 03 jun. 2025.

JUPYTER. Jupyter Documentation. Project Jupyter, 2025. Disponível em: https://docs.jupyter.org/en/latest/. Acesso em: 03 jun. 2025.

MATPLOTLIB. matplotlib.pyplot. Matplotlib, 2022. Disponível em: https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html. Acesso em: 03 jun. 2025.

NUMPY. NumPy Documentation. NumPy, 2025. Disponível em: https://numpy.org/doc/stable/. Acesso em: 03 jun. 2025.

OPENPYXL. openpyxl documentation. OpenPyXL, 2025. Disponível em: https://openpyxl.readthedocs.io/en/stable/#documentation. Acesso em: 03 jun. 2025.

PANDAS. pandas.DataFrame.dropna. Pandas Documentation, 2025. Disponível em: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html. Acesso em: 03 jun. 2025.

PANDAS. pandas.DataFrame.groupby. Pandas Documentation, 2025. Disponível em: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html. Acesso em: 03 jun. 2025.

PANDAS. pandas.read_excel. Pandas Documentation, 2025. Disponível em: https://pandas.pydata.org/docs/reference/api/pandas.read_excel.html. Acesso em: 03 jun. 2025.

PANDAS. pandas.to_datetime. Pandas Documentation, 2025. Disponível em: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_datetime.html. Acesso em: 03 jun. 2025.

PANDAS. Time series / date functionality. Pandas Documentation, 2025. Disponível em: https://pandas.pydata.org/docs/user_guide/timeseries.html#dateoffset-objects. Acesso em: 03 jun. 2025.

PANDAS. User Guide. Pandas Documentation, 2025. Disponível em: https://pandas.pydata.org/docs/. Acesso em: 03 jun. 2025.

SCIKIT-LEARN. Scikit-learn: Machine Learning in Python. Scikit-learn, 2025. Disponível em: https://scikit-learn.org/stable/. Acesso em: 03 jun. 2025.