PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS ESCOLA POLITÉCNICA E DE ARTES GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

AED1 – AI

RODRIGO F N VIEIRA

GOIÂNIA

2025

AED – INTELIGÊNCIA ARTIFICAL

Aluno: Rodrigo F N Vieira

Professor: Clarimar J. Coelho

Respostas:

A) Nesta etapa, foram importadas bibliotecas fundamentais para manipulação de dados (**pandas, numpy**) e para visualização gráfica (**matplotlib.pyplot e seaborn**), necessárias para a análise exploratória e implementação dos modelos de regressão.

```
# QUESTÃO (a) - IMPORTAÇÃO DE BIBLIOTECAS
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

B) Foi utilizado **pd.read_csv()** para ler o arquivo **beer_consumption.csv**, carregando os dados em um DataFrame chamado df, permitindo acesso facilitado às colunas e linhas da base de dados.

```
# QUESTÃO (b) - LEITURA DOS DADOS
df = pd.read_csv("C:/Users/didig/Downloads/Nova pasta/beer_consuption.csv")
```

C) A função **df.head()** exibe as primeiras 5 linhas do DataFrame, possibilitando a inspeção inicial dos dados, como nomes das colunas, tipos de valores e estrutura geral.

	Data	Temperatura Media (C)	Temperatura Minima (C)	Temperatura Maxima (C)	Precipitacao (mm)	Final de Semana	Consumo de cerveja (litros)
0	01/01/2015	27.30	23.9	32.5	0.0	0	25.461
1	02/01/2015	27.02	24.5	33.5	0.0	0	28.972
2	03/01/2015	24.82	22.4	29.9	0.0	1	30.814
3	04/01/2015	23.98	21.5	28.6	1.2	1	29.799
4	05/01/2015	23.82	21.0	28.3	0.0	0	28.900

D) A função **df.tail()** mostra as últimas 5 linhas, útil para verificar possíveis problemas no fim do arquivo ou padrões que aparecem apenas ao final da base de dados.

	Data	Temperatura Media (C)	Temperatura Minima (C)	Temperatura Maxima (C)	Precipitacao (mm)	Final de Semana	Consumo de cerveja (litros)
360	27/12/2015	24.00	21.1	28.2	13.6	1	32.307
361	28/12/2015	22.64	21.1	26.7	0.0	0	26.095
362	29/12/2015	21.68	20.3	24.1	10.3	0	22.309
363	30/12/2015	21.38	19.3	22.4	6.3	0	20.467
364	31/12/2015	24.76	20.2	29.0	0.0	0	22.446

E) O comando **df.shape** retorna o número de linhas e colunas da base. O resultado mostra que há 365 linhas (dias do ano) e 7 variáveis, como esperado.

```
Dimensão: (365, 7)
```

F) df.isnull().sum() contabiliza valores ausentes em cada coluna. O resultado confirma que não há dados faltando na base, o que dispensa tratamento de valores nulos.

```
Data                          0
Temperatura Media (C)         0
Temperatura Minima (C)        0
Temperatura Maxima (C)        0
Precipitacao (mm)             0
Final de Semana               0
Consumo de cerveja (litros)   0
dtype: int64
```

G) O comando **df.dtypes** identifica os tipos de dados em cada coluna (float, int ou object), fundamental para validar se os dados estão prontos para análises numéricas e modelagem.

```
Data                          object
Temperatura Media (C)        float64
Temperatura Minima (C)       float64
Temperatura Maxima (C)       float64
Precipitacao (mm)            float64
Final de Semana                int64
Consumo de cerveja (litros)  float64
dtype: object
```

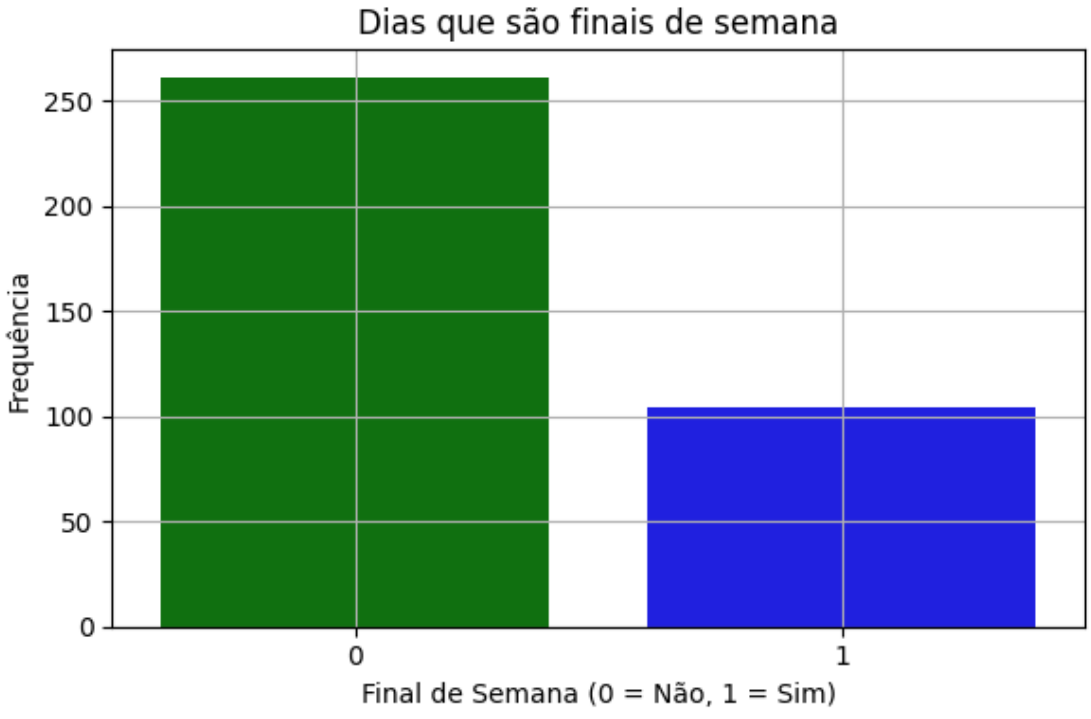
H) df.corr() calcula a matriz de correlação de Pearson entre variáveis numéricas. Permite identificar relações lineares, como o impacto da temperatura sobre o consumo de cerveja.

	Temperatura Media (C)	Temperatura Minima (C)	Temperatura Maxima (C)	Precipitacao (mm)	Final de Semana	Consumo de cerveja (litros)
Temperatura Media (C)	1.000000	0.862752	0.922513	0.024416	-0.050803	0.574615
Temperatura Minima (C)	0.862752	1.000000	0.672929	0.098625	-0.059534	0.392509
Temperatura Maxima (C)	0.922513	0.672929	1.000000	-0.049305	-0.040258	0.642672
Precipitacao (mm)	0.024416	0.098625	-0.049305	1.000000	0.001587	-0.193784
Final de Semana	-0.050803	-0.059534	-0.040258	0.001587	1.000000	0.505981
Consumo de cerveja (litros)	0.574615	0.392509	0.642672	-0.193784	0.505981	1.000000

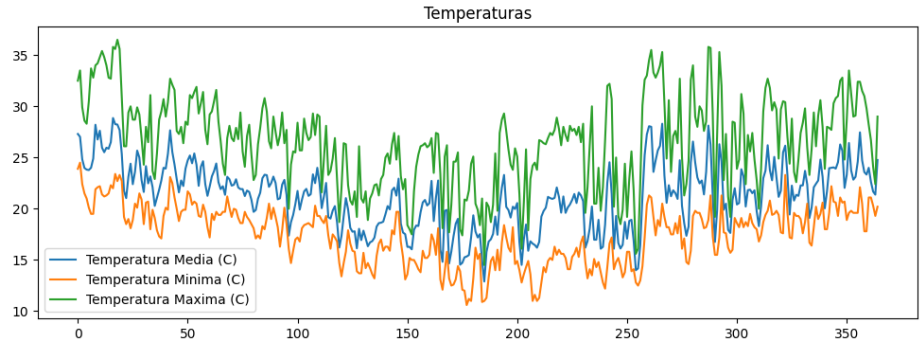
I) df.describe() fornece estatísticas como média, mediana, desvio padrão, valores máximos e mínimos, ajudando a compreender a distribuição e variação dos dados.

	Temperatura Media (C)	Temperatura Minima (C)	Temperatura Maxima (C)	Precipitacao (mm)	Final de Semana	Consumo de cerveja (litros)
count	365.000000	365.000000	365.000000	365.000000	365.000000	365.000000
mean	21.226356	17.461370	26.611507	5.196712	0.284932	25.401367
std	3.180108	2.826185	4.317366	12.417844	0.452001	4.399143
min	12.900000	10.600000	14.500000	0.000000	0.000000	14.343000
25%	19.020000	15.300000	23.800000	0.000000	0.000000	22.008000
50%	21.380000	17.900000	26.900000	0.000000	0.000000	24.867000
75%	23.280000	19.600000	29.400000	3.200000	1.000000	28.631000
max	28.860000	24.500000	36.500000	94.800000	1.000000	37.937000

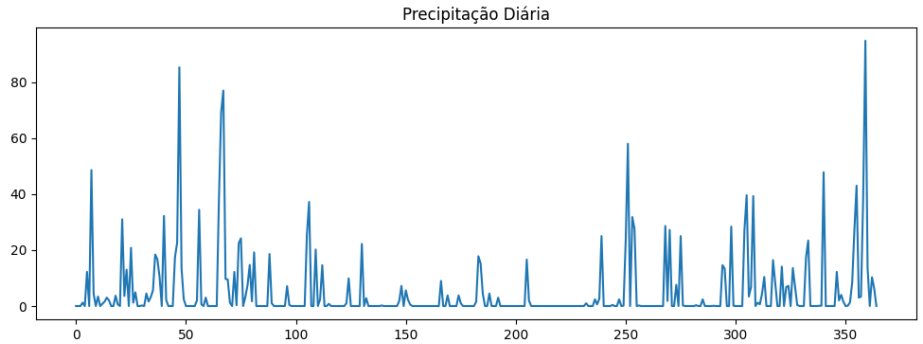
J) Um gráfico de barras mostra a frequência dos dias que são finais de semana (1) e dos que não são (0). O gráfico confirma que a maioria dos dias não são finais de semana.



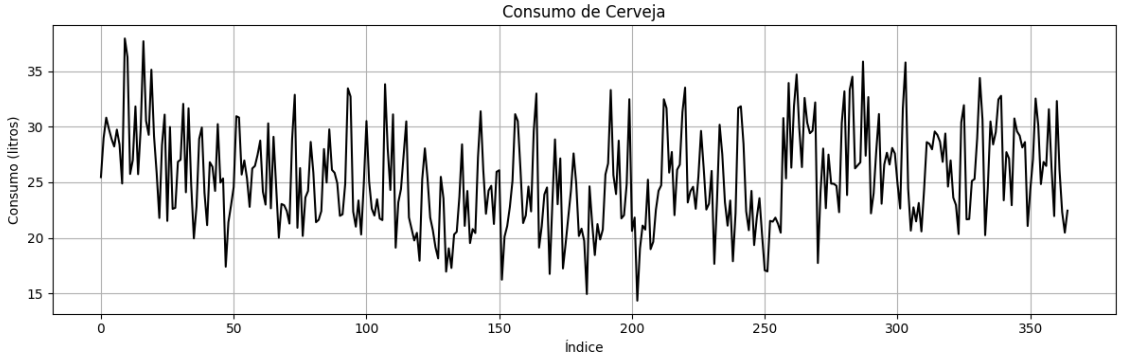
k) Foi gerado um gráfico de linha com as temperaturas média, mínima e máxima ao longo do ano, permitindo visualizar padrões sazonais e flutuações térmicas.



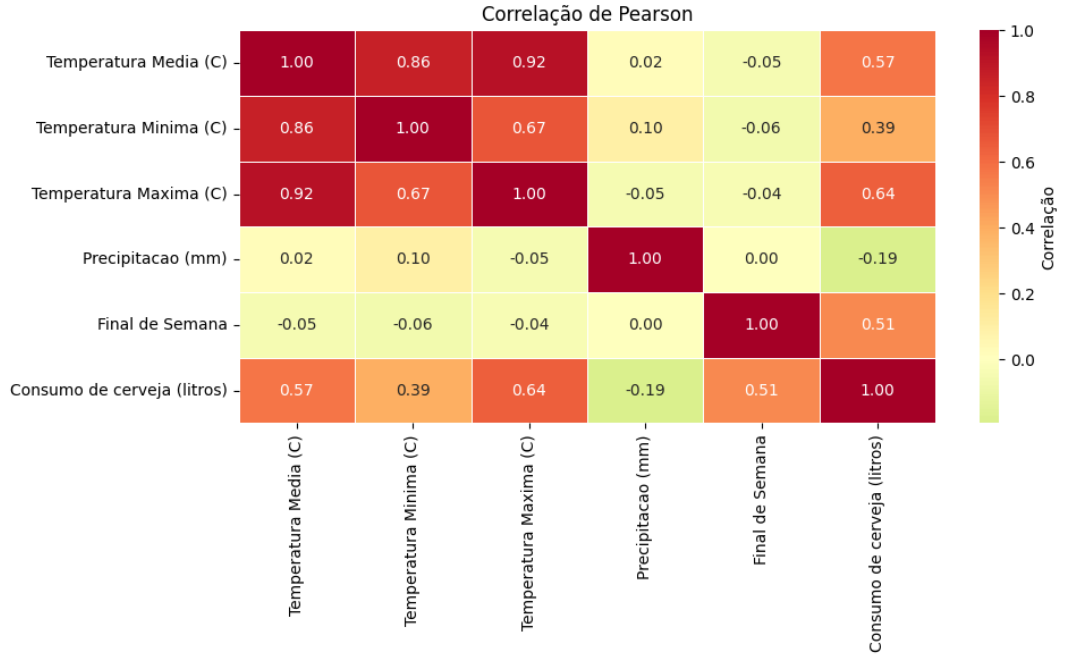
L) Mostra a variação da precipitação diária (em mm) ao longo do ano. O gráfico ajuda a identificar períodos mais chuvosos, que podem influenciar o consumo de cerveja.



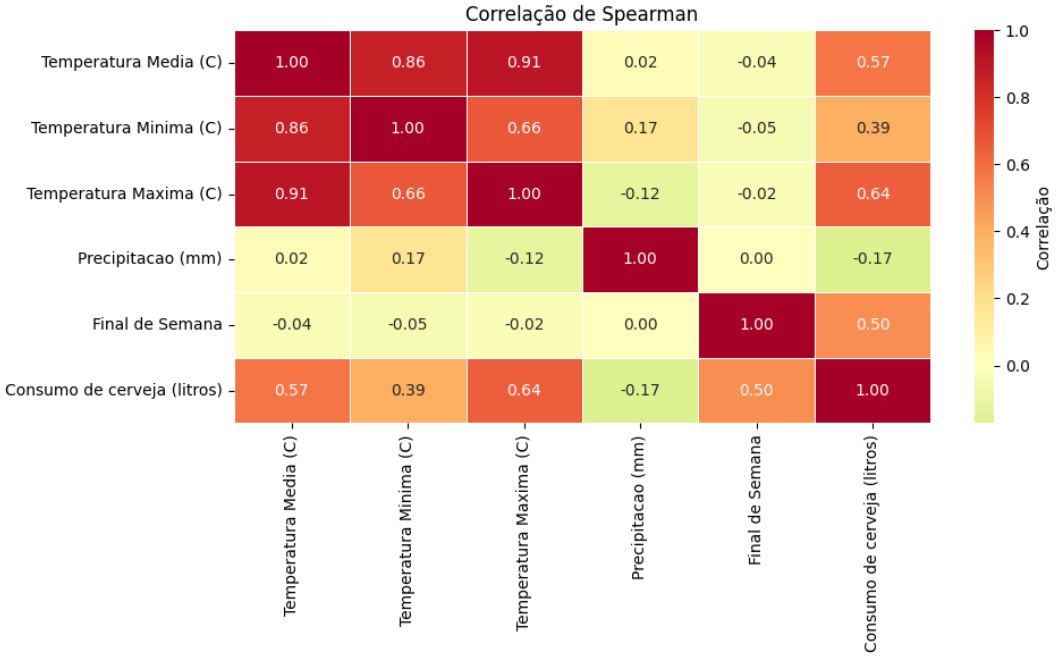
M) Representa visualmente o consumo de cerveja em litros por dia. Ajuda a identificar tendências e oscilações, como maior consumo em dias mais quentes.



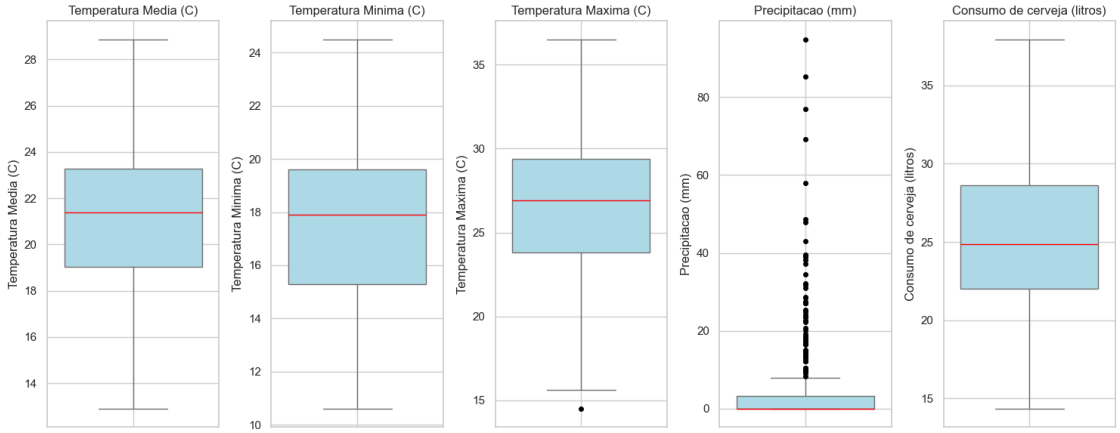
N) Um heatmap de correlação de Pearson mostra graficamente a força e direção das correlações lineares entre as variáveis, com destaque para a relação entre temperatura e consumo.



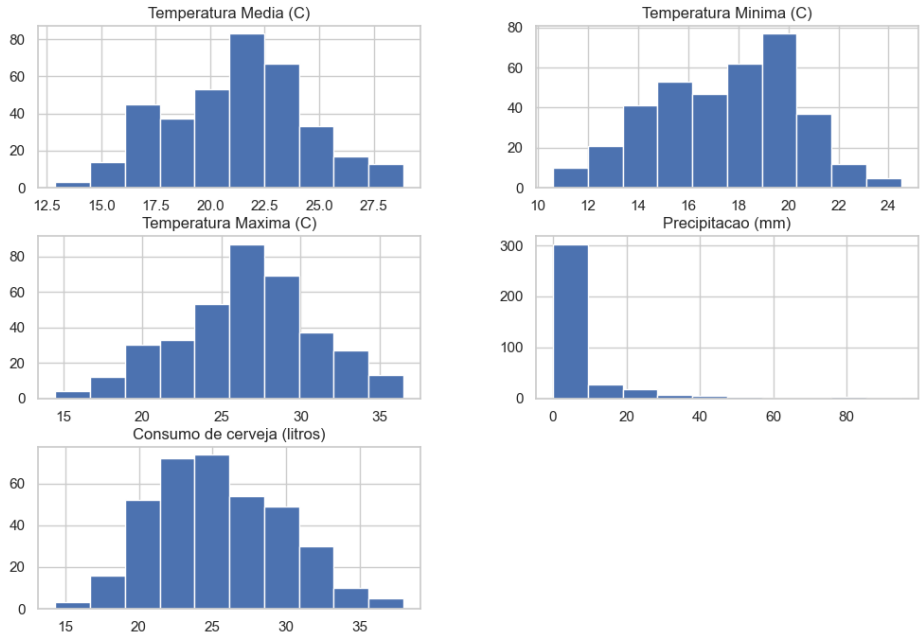
O) Heatmap similar ao anterior, mas baseado no coeficiente de Spearman, que capta relações monotônicas (não necessariamente lineares). Útil se os dados tiverem outliers ou não-linearidades.



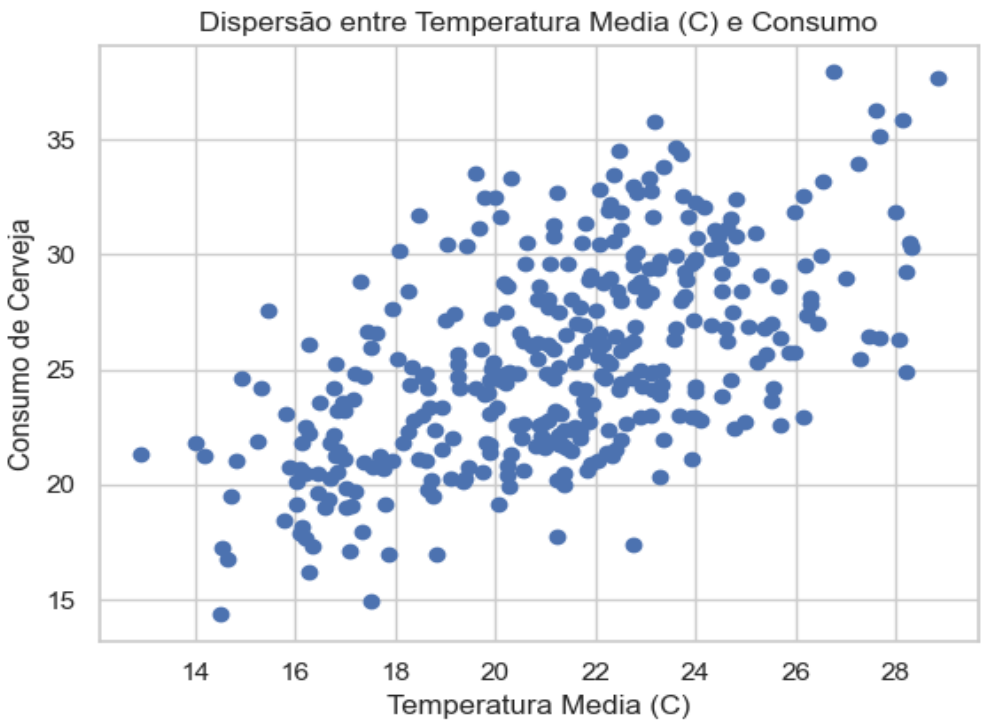
P) Os boxplots mostram a distribuição de cada variável e possíveis valores extremos (outliers). Permitem observar a dispersão e assimetria dos dados.

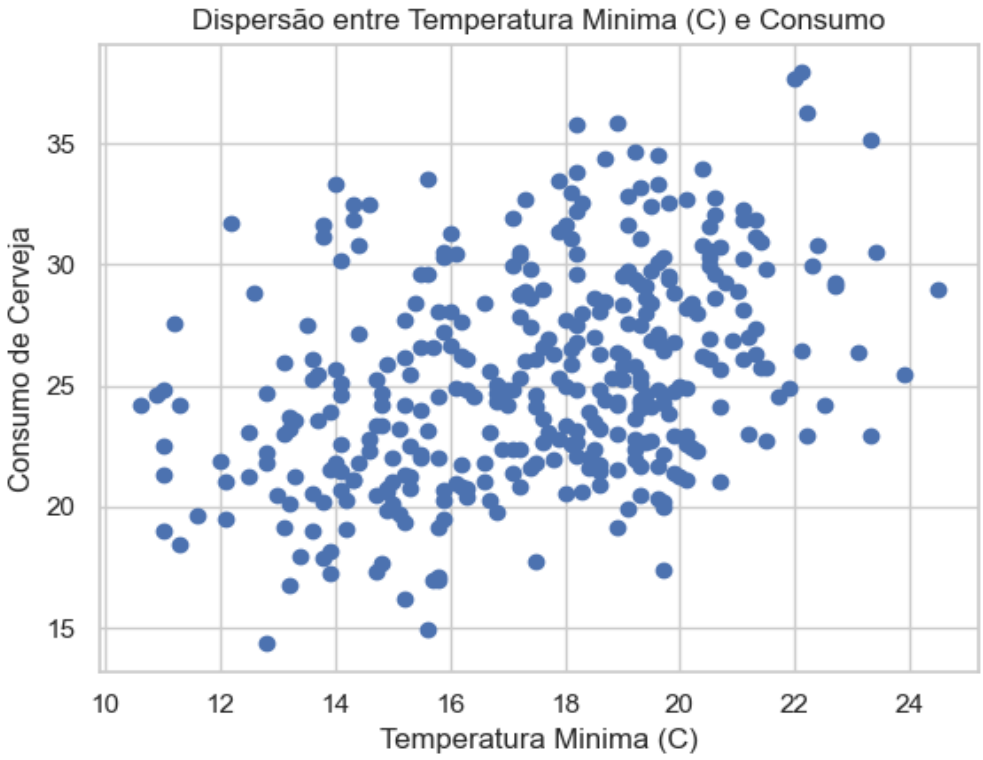


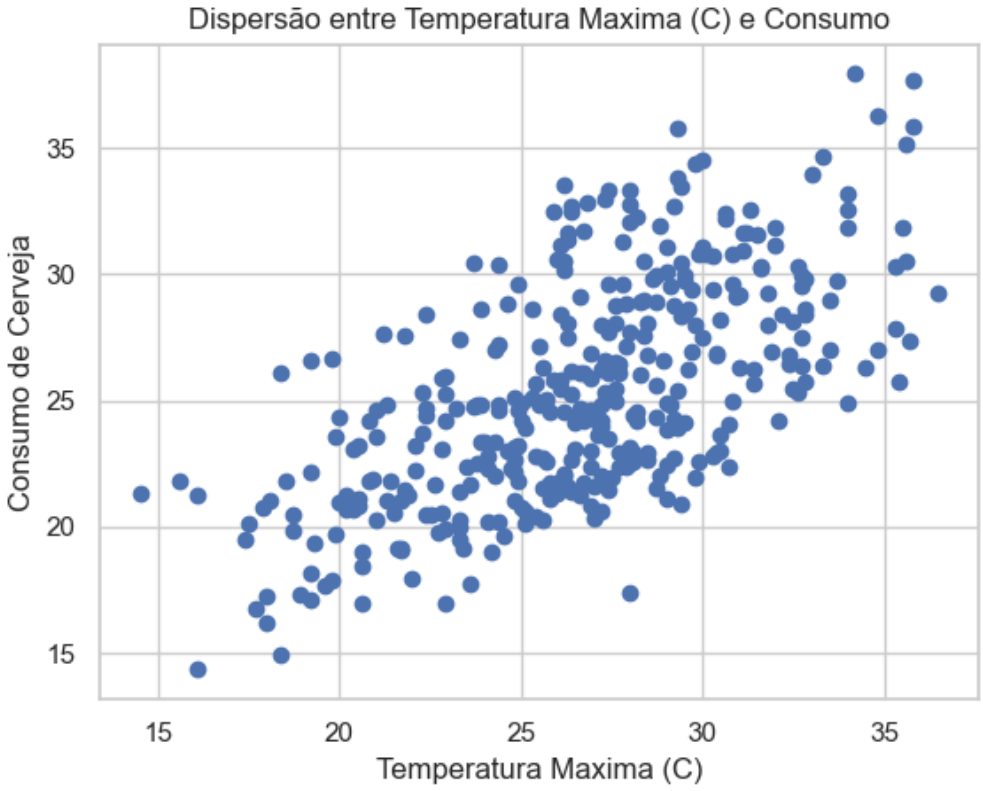
Q) Exibem a distribuição das variáveis numéricas. Os histogramas mostram a frequência dos valores, revelando padrões como normalidade, assimetria ou presença de valores extremos.

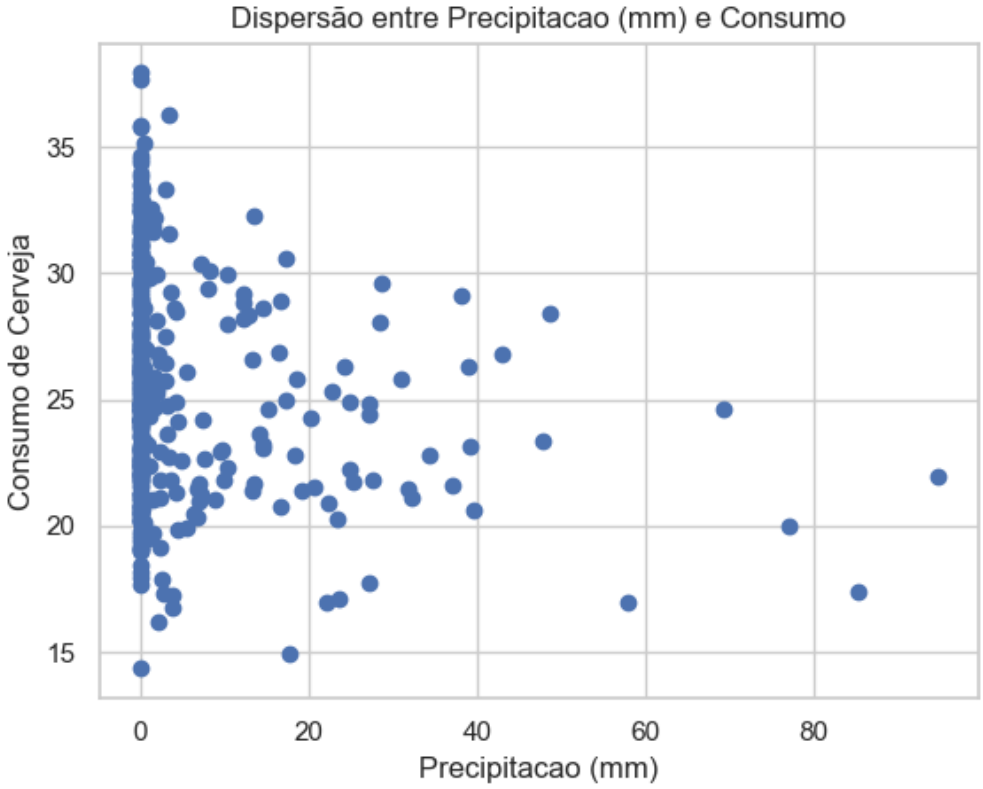


R) Exibem a relação entre o consumo de cerveja e as demais variáveis. Permite avaliar visualmente a existência de correlação (ex: maior temperatura tende a significar maior consumo).






Dispersão entre Temperatura Maxima (C) e Consumo
Consumo de Cerveja
35
30
25
20
15
15
20
25
30
35
Temperatura Maxima (C)


Dispersão entre Precipitacao (mm) e Consumo
Consumo de Cerveja
35
30
25
20
15
0
20
40
60
80
Precipitacao (mm)

S) Foram implementados dois modelos:

- (a) MMQ (Mínimos Quadrados): cálculo direto dos coeficientes usando álgebra matricial.
- (b) Gradiente Descendente: aproximação iterativa dos coeficientes.
- (c) Comparação: ambos modelos geram resultados semelhantes. O MMQ é exato, o gradiente pode variar dependendo da taxa de aprendizado e número de épocas.

T) Foram calculadas métricas para avaliar os modelos:

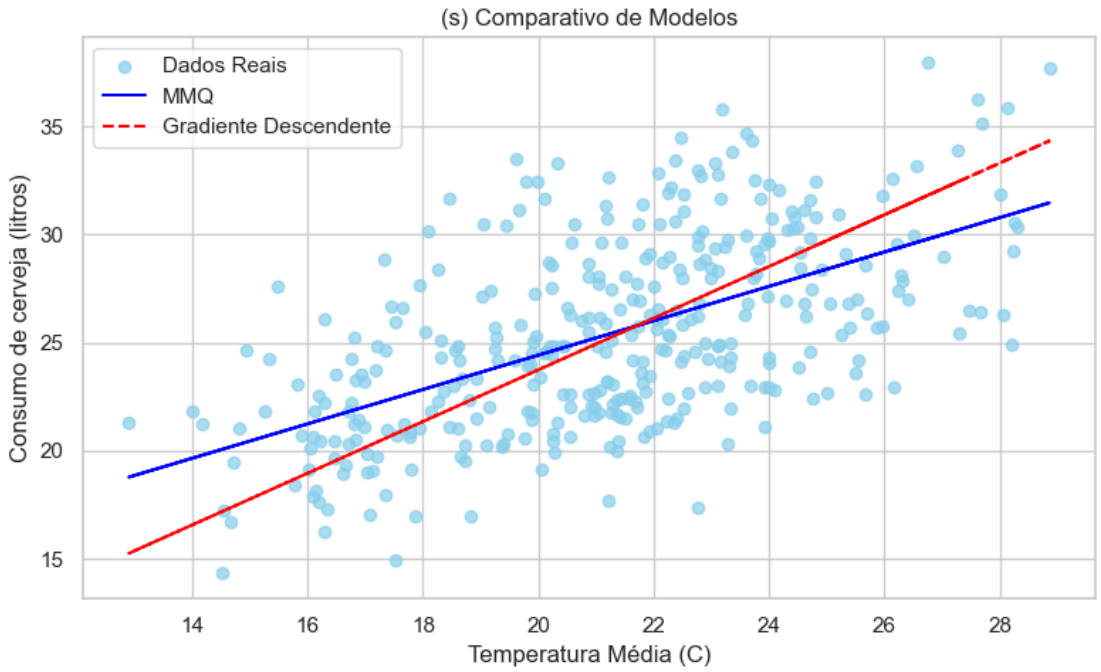
- R^2 e R^2 ajustado: medem o quanto o modelo explica a variabilidade dos dados.
- MSE, RMSE: erro médio quadrático e sua raiz.
- MAE, MAPE: erro absoluto médio e percentual.
- RMSLE: erro logarítmico, útil para evitar penalização excessiva de grandes desvios.

```
Coeficientes (MMQ): [ 6.44469636  0.03079559  0.65600076 -0.01903491 -0.05746938  5.18318073]

Coeficientes (Gradiente Descendente): [ 0.3343196  -0.08334787  0.79952112  0.31549159 -0.05588016  0.60218432]

--- MÉTRICAS PARA MMQ ---
R²: 0.7226
R² ajustado: 0.7188
MSE: 5.3527
RMSE: 2.3136
MAE: 1.9640
MAPE: 7.82%
RMSLE: 0.0873

--- MÉTRICAS PARA GRADIENTE DESCENDENTE ---
R²: 0.4245
R² ajustado: 0.4165
MSE: 11.1065
RMSE: 3.3326
MAE: 2.7059
MAPE: 10.70%
RMSLE: 0.1262
```



Código:

```
# QUESTÃO (a) - IMPORTAÇÃO DE BIBLIOTECAS
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# QUESTÃO (b) - LEITURA DOS DADOS
df = pd.read_csv("C:/Users/didig/Downloads/Nova
pasta/beer_consuption.csv")

# QUESTÕES (c) a (g) - EXPLORAÇÃO INICIAL
print(df.head())     # (c)
print(df.tail())     # (d)
print("Dimensão:", df.shape)  # (e)
print(df.isnull().sum())      # (f)
print(df.dtypes)              # (g)

# QUESTÃO (h) - CORRELAÇÃO DE PEARSON
print(df.corr(numeric_only=True))

# QUESTÃO (i) - ESTATÍSTICAS DESCRITIVAS
print(df.describe())

# QUESTÃO (j) - GRÁFICO DE FINAIS DE SEMANA
plt.figure(figsize=(6,4))
cores = {'0': 'green', '1': 'blue'}
df['Final de Semana'] = df['Final de Semana'].astype(int)
```

```
sns.countplot(x='Final de Semana', data=df, palette=cores)
plt.title("Dias que são finais de semana")
plt.xlabel("Final de Semana (0 = Não, 1 = Sim)")
plt.ylabel("Frequência")
plt.grid(True)
plt.tight_layout()
plt.show()

# QUESTÃO (k) - GRÁFICO DAS TEMPERATURAS
df[['Temperatura Media (C)', 'Temperatura Minima (C)', 'Temperatura
Maxima (C)']].plot(figsize=(12,4))
plt.title("Temperaturas")
plt.show()

# QUESTÃO (l) - GRÁFICO DA PRECIPITAÇÃO
df['Precipitacao (mm)'].plot(figsize=(12,4))
plt.title("Precipitação Diária")
plt.show()

# QUESTÃO (m) - GRÁFICO DO CONSUMO
df['Consumo de cerveja (litros)'].plot(figsize=(12,4), color='black')
plt.title("Consumo de Cerveja")
plt.xlabel("Índice")
plt.ylabel("Consumo (litros)")
plt.grid(True)
plt.tight_layout()
plt.show()

# QUESTÃO (n) - HEATMAP PEARSON
plt.figure(figsize=(10, 6))
sns.heatmap(
    df.corr(numeric_only=True),
    annot=True,
    fmt=".2f",
    cmap='RdYlGn_r',  # <- paleta invertida
    center=0,
    linewidths=0.5,
    linecolor='white',
    cbar_kws={'label': 'Correlação'}
)
plt.title("Correlação de Pearson")
plt.tight_layout()
plt.show()

# QUESTÃO (o) - HEATMAP SPEARMAN
plt.figure(figsize=(10, 6))
sns.heatmap(
    df.corr(method='spearman', numeric_only=True),
    annot=True,
```

```
    fmt=".2f",
    cmap='RdYlGn_r',
    center=0,
    linewidths=0.5,
    linecolor='white',
    cbar_kws={'label': 'Correlação'}
)
plt.title("Correlação de Spearman")
plt.tight_layout()
plt.show()

# QUESTÃO (p) - BOXPLOTS AJUSTADOS
variaveis = ['Temperatura Media (C)', 'Temperatura Minima (C)',
'Temperatura Maxima (C)',
             'Precipitacao (mm)', 'Consumo de cerveja (litros)']
plt.figure(figsize=(15, 6))
sns.set(style="whitegrid")
for i, var in enumerate(variaveis, 1):
    plt.subplot(1, 5, i)
    sns.boxplot(
        y=df[var],
        color='skyblue',
        boxprops=dict(facecolor='lightblue'),
        medianprops=dict(color='red'),
        flierprops=dict(marker='o', markersize=4,
markerfacecolor='black', markeredgecolor='black')
    )
    plt.title(var)
    plt.grid(True)
plt.tight_layout()
plt.show()

# QUESTÃO (q) - HISTOGRAMAS
df[variaveis].hist(figsize=(12,8))
plt.show()

# QUESTÃO (r) - GRÁFICOS DE DISPERSÃO
for col in variaveis[:-1]:
    plt.scatter(df[col], df['Consumo de cerveja (litros)'])
    plt.xlabel(col)
    plt.ylabel("Consumo de Cerveja")
    plt.title(f"Dispersão entre {col} e Consumo")
    plt.show()

# QUESTÃO (s) - REGRESSÃO LINEAR MÚLTIPLA
print("""\n
(s) Construir um modelo de regressão linear múltipla em Python
    (a) Usando MMQ
    (b) Gradiente descendente
```

```
    (c) Comparar os resultados.
""")

X = df[['Temperatura Media (C)', 'Temperatura Maxima (C)', 'Temperatura
Minima (C)',
        'Precipitacao (mm)', 'Final de Semana']].values
y = df['Consumo de cerveja (litros)'].values
X_b = np.c_[np.ones((X.shape[0], 1)), X]

# (s.a) MMQ
beta = np.linalg.inv(X_b.T @ X_b) @ X_b.T @ y
print("\nCoeficientes (MMQ):", beta)

# (s.b) Gradiente Descendente
def gradient_descent(X, y, alpha=0.0001, epochs=1000):
    m = len(y)
    X_b = np.c_[np.ones((m, 1)), X]
    theta = np.random.randn(X_b.shape[1])
    for _ in range(epochs):
        gradients = 2/m * X_b.T @ (X_b @ theta - y)
        theta -= alpha * gradients
    return theta

theta = gradient_descent(X, y)
print("\nCoeficientes (Gradiente Descendente):", theta)

# QUESTÃO (t) - CÁLCULO DAS MÉTRICAS
def calcular_metricas(y_true, y_pred, X):
    n = len(y_true)
    p = X.shape[1]
    mse = np.mean((y_true - y_pred)**2)
    rmse = np.sqrt(mse)
    mae = np.mean(np.abs(y_true - y_pred))
    mape = np.mean(np.abs((y_true - y_pred) / y_true)) * 100
    rmsle = np.sqrt(np.mean((np.log1p(y_pred) - np.log1p(y_true))**2))
    r2 = 1 - np.sum((y_true - y_pred)**2) / np.sum((y_true -
np.mean(y_true))**2)
    r2_adj = 1 - (1 - r2) * ((n - 1) / (n - p - 1))

    print(f"R²: {r2:.4f}")
    print(f"R² ajustado: {r2_adj:.4f}")
    print(f"MSE: {mse:.4f}")
    print(f"RMSE: {rmse:.4f}")
    print(f"MAE: {mae:.4f}")
    print(f"MAPE: {mape:.2f}%")
    print(f"RMSLE: {rmsle:.4f}")

print("\n--- MÉTRICAS PARA MMQ ---")
y_pred_mmq = X_b @ beta
```

```
calcular_metricas(y, y_pred_mmq, X)

print("\n--- MÉTRICAS PARA GRADIENTE DESCENDENTE ---")
y_pred_gd = X_b @ theta
calcular_metricas(y, y_pred_gd, X)

# QUESTÃO (s.c) - GRÁFICO DE REGRESSÃO USANDO TEMPERATURA MÉDIA (C)
x_plot = df['Temperatura Media (C)'].values
x_plot_reshape = x_plot.reshape(-1, 1)
x_plot_b = np.c_[np.ones((x_plot.shape[0], 1)), x_plot]

# MMQ com 1 variável
beta_simples = np.linalg.inv(x_plot_b.T @ x_plot_b) @ x_plot_b.T @ y
y_pred_mmq_simples = x_plot_b @ beta_simples

# GD com 1 variável
def gd_simples(x, y, alpha=0.0001, epochs=1000):
    m = len(y)
    x_b = np.c_[np.ones((m, 1)), x]
    theta = np.random.randn(x_b.shape[1])
    for _ in range(epochs):
        gradients = 2/m * x_b.T @ (x_b @ theta - y)
        theta -= alpha * gradients
    return theta

theta_simples = gd_simples(x_plot_reshape, y)
y_pred_gd_simples = x_plot_b @ theta_simples

plt.figure(figsize=(8, 5))
plt.scatter(x_plot, y, color='skyblue', alpha=0.7, label='Dados Reais')
plt.plot(x_plot, y_pred_mmq_simples, color='blue', label='MMQ')
plt.plot(x_plot, y_pred_gd_simples, color='red', linestyle='--',
label='Gradiente Descendente')
plt.xlabel("Temperatura Média (C)")
plt.ylabel("Consumo de cerveja (litros)")
plt.title("(s) Comparativo de Modelos")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```