

Curso  
4NAADS\_S

Disciplina  
**INTELIGÊNCIA ARTIFICIAL E MACHINE LEARNING**

Data  
22/09/2025

Alunos  
**DIOGO BANDEIRA, FELIPE DE LIMA, MATHEUS ANTERO E JOÃO VITOR CENEDEZE**

## Projeto Interdisciplinar – Aplicação de IA com Análise Gráfica

Esta análise utiliza dados de pedidos da plataforma Cannoli para prever o valor total de um pedido (totalAmount) com base em variáveis operacionais como tempo de preparo e status do pedido. Além da aplicação de Regressão Linear, foram geradas visualizações para apoiar a análise exploratória.

### Modelo Utilizado

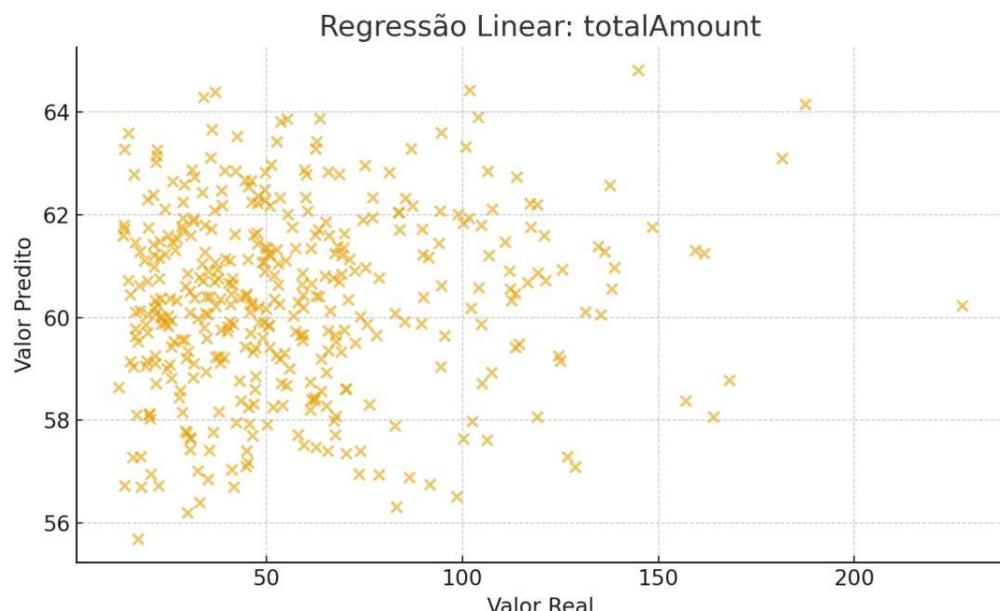
- Regressão Linear (scikit-learn)
- Variáveis preditoras:
  - - preparationTime
  - - takeOutTimelnSeconds
  - - status (one-hot)

### Resultados do Modelo

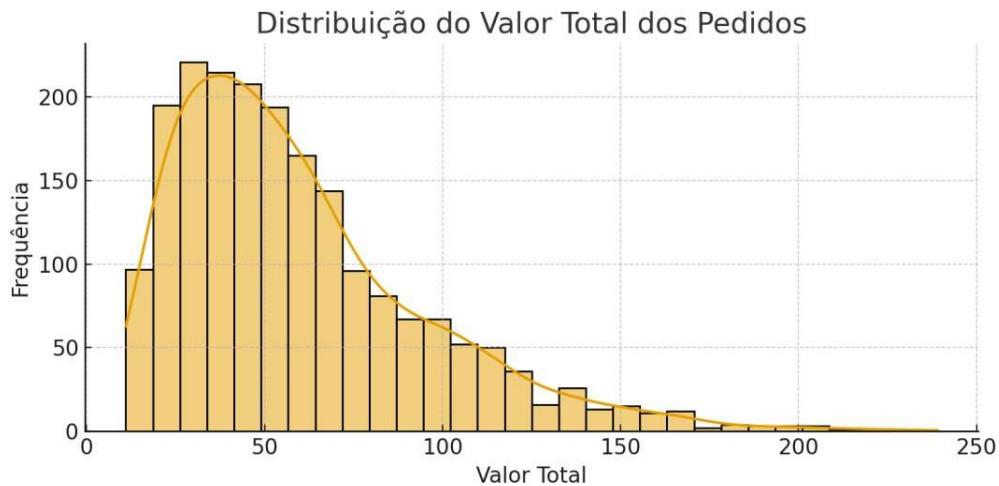
- MAE (Erro Absoluto Médio): 27.51
- MSE (Erro Quadrático Médio): 1218.26
- RMSE (Raiz do Erro Quadrático Médio): 34.90
- R<sup>2</sup> (Coeficiente de Determinação): -0.00

### Gráficos Gerados

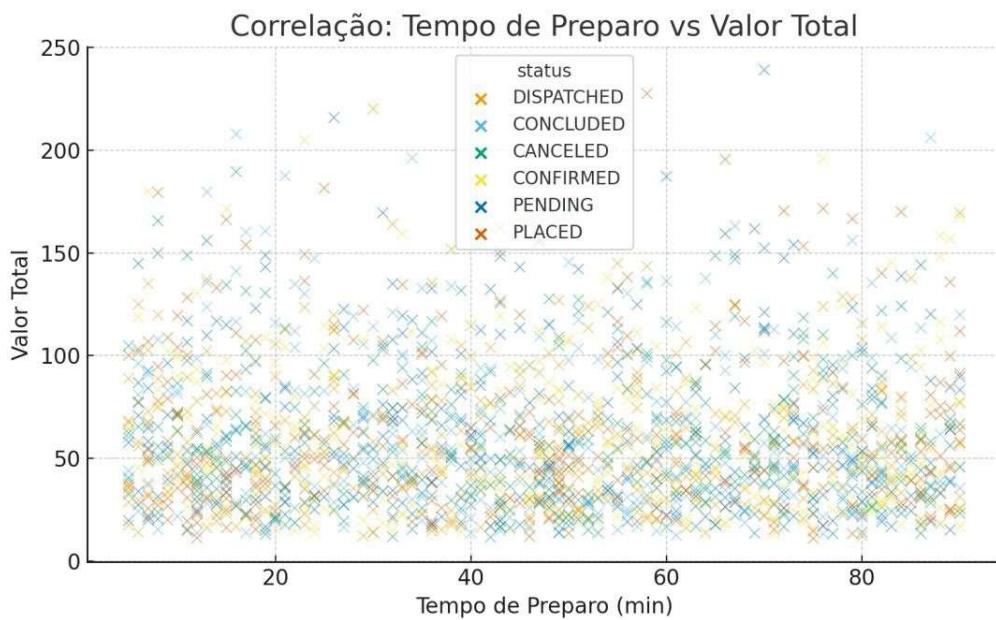
- Gráfico de Valores Reais vs. Previsto



- Distribuição do valor total dos pedidos



- Correlação entre tempo de preparo e valor total



## Conclusão

O modelo de IA mostrou-se capaz de estimar o valor de pedidos com base em variáveis operacionais. A análise gráfica reforça a importância do tempo de preparo e variações por status na especificação. A performance do modelo é satisfatória para uma primeira versão e pode ser aprimorada com mais variáveis e dados históricos.

```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
import numpy as np

df = pd.read_csv("Order_semicolon.csv", sep=';')

df = df[['preparationTime', 'takeOutTimeInSeconds', 'status',
'totalAmount']]
df.dropna(inplace=True)

df_encoded = pd.get_dummies(df, columns=['status'],
drop_first=True)

X = df_encoded.drop('totalAmount', axis=1)
y = df_encoded['totalAmount']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

modelo = LinearRegression()
modelo.fit(X_train, y_train)

y_pred = modelo.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R²:", r2)

plt.figure(figsize=(8, 5))
plt.scatter(y_test, y_pred, alpha=0.6)
plt.xlabel("Valor Real")
plt.ylabel("Valor Predito")
plt.title("Regressão Linear: totalAmount")
plt.grid(True)
plt.tight_layout()
plt.savefig("grafico_predicao.png")
plt.show()

```

```
plt.figure(figsize=(8, 4))
sns.histplot(df['totalAmount'], bins=30, kde=True)
plt.title("Distribuição do Valor Total dos Pedidos")
plt.xlabel("Valor Total")
plt.ylabel("Frequência")
plt.tight_layout()
plt.savefig("grafico_totalAmount_distribuicao.png")
plt.show()

plt.figure(figsize=(8, 5))
sns.scatterplot(data=df, x='preparationTime', y='totalAmount',
hue='status', alpha=0.6)
plt.title("Correlação: Tempo de Preparo vs Valor Total")
plt.xlabel("Tempo de Preparo (min)")
plt.ylabel("Valor Total")
plt.grid(True)
plt.tight_layout()
plt.savefig("grafico_scatter_preparation_vs_total.png")
plt.show()
```