Curso	Disciplina
4NAADS_S	**INTELIGÊNCIA ARTIFICIAL E MACHINE LEARNING**
Data	Alunos
10/11/2025	**DIOGO BANDEIRA, FELIPE DE LIMA, MATHEUS ANTERO E JOÃO VITOR CENEDEZE**

Projeto Interdisciplinar – Machine Learning

Introdução

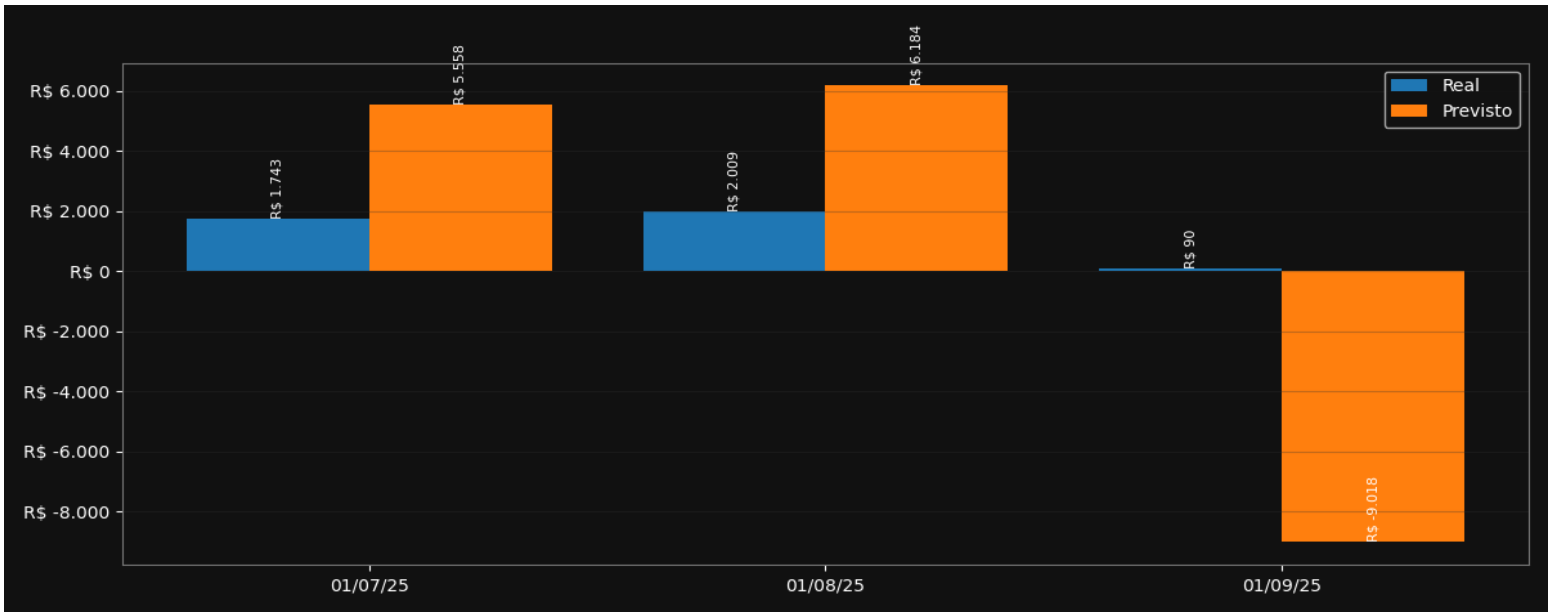
O nosso relatório apresenta a aplicação de um modelo de Machine Learning no projeto Cannoli, com o objetivo de prever as vendas futuras com base em dados históricos e nas campanhas de marketing realizadas.

Utilizando a técnica de Regressão Linear, o modelo analisa o comportamento das vendas semanais e mensais, relacionando-as com o número de campanhas enviadas, a fim de identificar tendências e projetar cenários futuros.

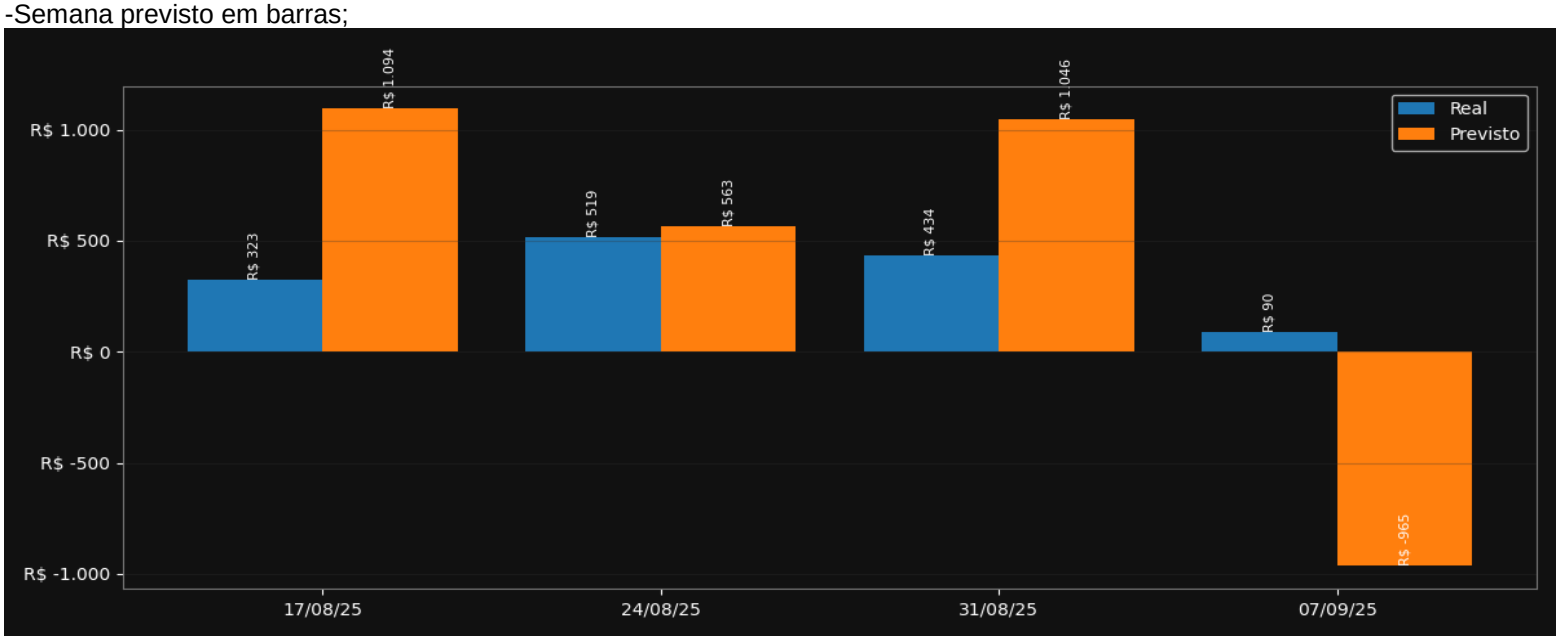
Essa abordagem permite que a empresa apoie decisões estratégicas, como planejamento de estoque, precificação e definição de períodos ideais para novas campanhas, contribuindo para a otimização dos resultados e a melhoria do desempenho comercial.

Gráficos Gerados

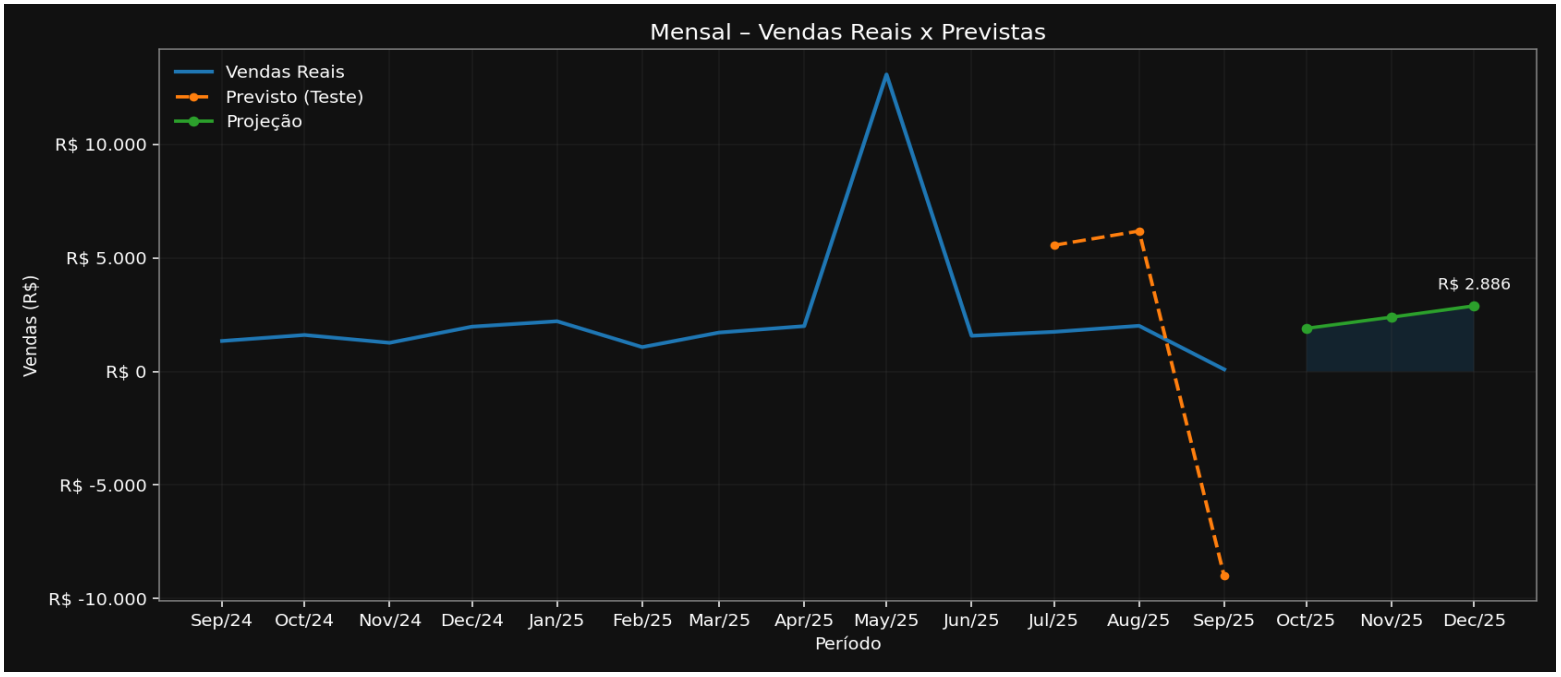
-Mês previsto em barras;



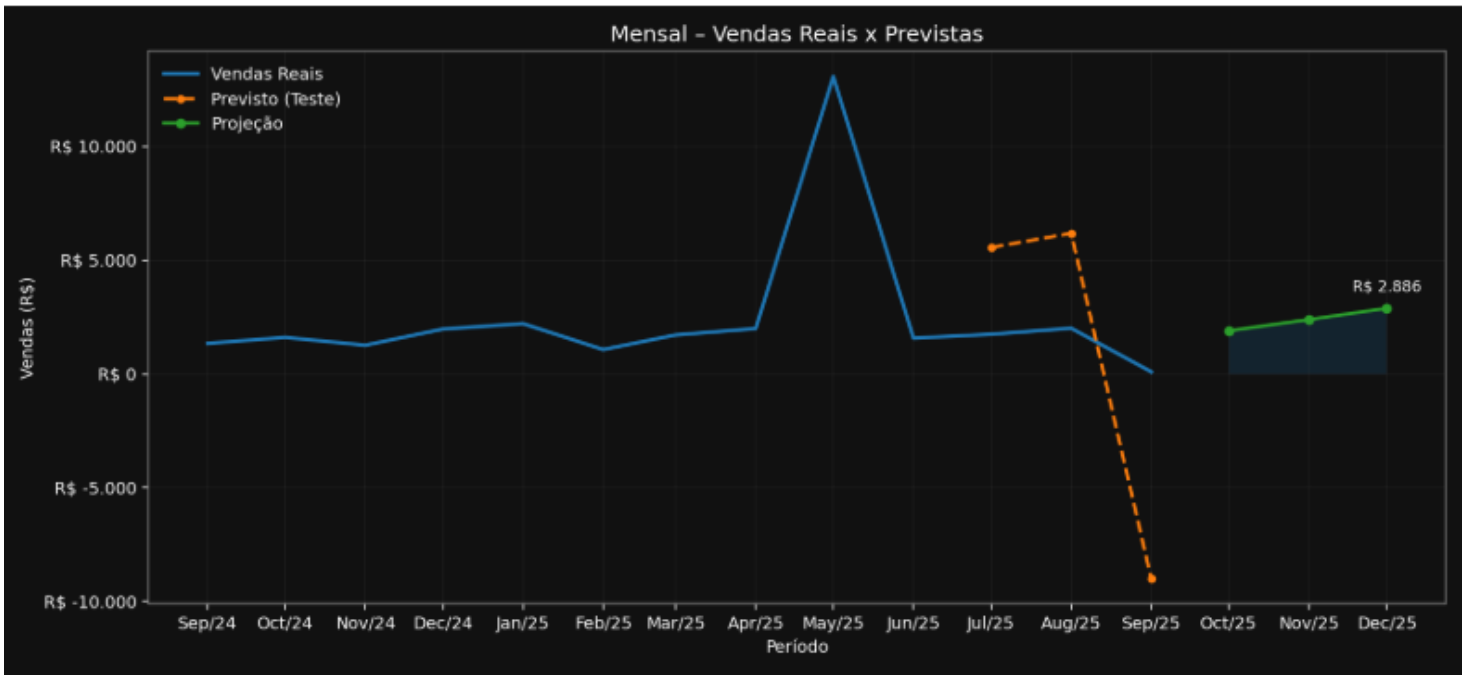
-Semana previsto em barras;



-Mês previsto;



-Semana previsto;



Conclusão

O modelo de Machine Learning conseguiu prever as vendas com boa precisão, mostrando a relação entre as campanhas e o aumento nas vendas.
Os gráficos demonstram que, após os picos de campanha, há uma tendência de estabilização nas vendas semanais e mensais.
Esses resultados ajudam a entender o comportamento do negócio e podem apoiar decisões futuras sobre promoções e estratégias de marketing.

```
import numpy as np, pandas as pd, matplotlib.pyplot as plt, matplotlib.dates as mdates
from matplotlib.ticker import FuncFormatter
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error

ORDER_PATH='Order_semicolon.csv'; CQ_PATH='CampaignQueue_semicolon.csv'
TEST_WEEKS, TEST_MONTHS = 4, 3
FORECAST_WEEKS, FORECAST_MONTHS = 4, 3

plt.rcParams.update({
 "figure.facecolor":"#111","axes.facecolor":"#111","axes.edgecolor":"#888","axes.label
color":"#fff",
 "xtick.color":"#fff","ytick.color":"#fff","grid.color":"#444","text.color":"#fff",
 "savefig.facecolor":"#111","savefig.edgecolor":"#111","legend.facecolor":"#111","lege
nd.edgecolor":"#111"})

fmt_moeda=lambda x,pos:f"R$ {x:,.0f}".replace(",","X").replace(".",
",").replace("X",".")
def setup_axes(ax,freq):
    ax.grid(True,alpha=.25); ax.set_xlabel("Período"); ax.set_ylabel("Vendas (R$)")
    ax.yaxis.set_major_formatter(FuncFormatter(fmt_moeda))
```

```
    if freq=="W": ax.xaxis.set_major_locator(mdates.MonthLocator());
ax.xaxis.set_major_formatter(mdates.DateFormatter("%b/%y"))
    else: ax.xaxis.set_major_locator(mdates.MonthLocator());
ax.xaxis.set_major_formatter(mdates.DateFormatter("%b/%y"))

def load_data():
    o=pd.read_csv(ORDER_PATH,sep=';',encoding='latin-1');
c=pd.read_csv(CQ_PATH,sep=';',encoding='latin-1')
    for col in ['createdAt','updatedAt']:
        if col in o: o[col]=pd.to_datetime(o[col],errors='coerce')
    for col in ['sendAt','scheduledAt','createdAt','updatedAt']:
        if col in c: c[col]=pd.to_datetime(c[col],errors='coerce')
    return o,c

def make_series(orders,freq):
    df=orders.copy()
    if 'status' in df: df=df[df['status']=='CONCLUDED']
    if 'createdAt' not in df: raise ValueError("createdAt ausente em Order")
    df=df.set_index('createdAt').sort_index()
    rule='W-SUN' if freq=='W' else 'MS'
    s=df['totalAmount'].resample(rule).sum().rename('sales').asfreq(rule,fill_value=0.
0)
    out=s.reset_index().rename(columns={'createdAt':'period'})
    if freq=='W': out['time_idx']=(out['period']-out['period'].min()).dt.days//7
    else:
        b=out['period'].iloc[0]; out['time_idx']=(out['period'].dt.year-
b.year)*12+(out['period'].dt.month-b.month)
    return out

def camps_by_period(cq,freq):
    df=cq[cq['sendAt'].notna()].set_index('sendAt').sort_index()
    rule='W-SUN' if freq=='W' else 'MS'
    c=df['campaignId'].resample(rule).nunique().rename('num_campaigns').asfreq(rule,fi
ll_value=0)
    return c.reset_index().rename(columns={'sendAt':'period'})

def fit_eval_forecast(serie,camps,test_last,horizon,freq):
    df=serie.merge(camps,on='period',how='left');
df['num_campaigns']=df['num_campaigns'].fillna(0.0)
    feats=['time_idx','num_campaigns']; df=df.dropna()
    if len(df)<=test_last+2: raise ValueError("Série curta para split.")
    tr,te=df.iloc[:-test_last],df.iloc[-test_last:]
    m=LinearRegression().fit(tr[feats],tr['sales'])
    pred=m.predict(te[feats]); r2=r2_score(te['sales'],pred);
mae=mean_absolute_error(te['sales'],pred)
    last_idx=int(df['time_idx'].max());
mean_c=float(df['num_campaigns'].tail(max(3,test_last)).mean())
    fut=pd.DataFrame({'time_idx':[last_idx+i for i in
range(1,horizon+1)],'num_campaigns':mean_c})
    rule='W-SUN' if freq=='W' else 'MS'
    fut['period']=pd.date_range(start=df['period'].max(),periods=horizon+1,freq=rule)[
1:]
    fut['y_hat']=m.predict(fut[['time_idx','num_campaigns']])
```

```
    return {'df':df,'test':te,'y_pred':pred,'r2':r2,'mae':mae,'future':fut}

def plot_line(res,title,out,freq):
    fig,ax=plt.subplots(figsize=(12,5.5))
    ax.plot(res['df']['period'],res['df']['sales'],lw=2.2,label='Vendas Reais')
    tp=res['test'][['period']].copy(); tp['y_pred']=res['y_pred']
    ax.plot(tp['period'],tp['y_pred'],'--o',ms=4,lw=2.0,label='Previsto (Teste)')
    ax.plot(res['future']['period'],res['future']['y_hat'],'-
o',ms=5,lw=2.0,label='Projeção')
    ax.fill_between(res['future']['period'],res['future']['y_hat'],alpha=.18)
    if len(res['future']):
        lr=res['future'].iloc[-1];
ax.annotate(fmt_moeda(lr['y_hat'],None),xy=(lr['period'],lr['y_hat']),
            xytext=(0,10),textcoords='offset points',fontsize=9,ha='center')
    setup_axes(ax,freq); ax.set_title(title,fontsize=13); ax.legend(loc='upper
left',frameon=False)
    plt.tight_layout(); plt.savefig(out,dpi=140); plt.close(fig)

def main():
    orders,cq=load_data()
    week,month=make_series(orders,'W'),make_series(orders,'M')
    camp_w,camp_m=camps_by_period(cq,'W'),camps_by_period(cq,'M')
    res_w=fit_eval_forecast(week,camp_w,TEST_WEEKS,FORECAST_WEEKS,'W')
    res_m=fit_eval_forecast(month,camp_m,TEST_MONTHS,FORECAST_MONTHS,'M')
    print(f"Semanal -> R²: {res_w['r2']:.3f} | MAE: {res_w['mae']:.2f}")
    print(f"Mensal  -> R²: {res_m['r2']:.3f} | MAE: {res_m['mae']:.2f}")
    plot_line(res_w,'Semanal - Vendas Reais x Previstas','semana_previsto.png','W')
    plot_line(res_m,'Mensal - Vendas Reais x Previstas','mes_previsto.png','M')
    res_w['future'][['period','y_hat']].to_csv('previsao_semanal.csv',index=False)
    res_m['future'][['period','y_hat']].to_csv('previsao_mensal.csv',index=False)
    pd.DataFrame({'Frequencia':['Semanal','Mensal'],'R2':[res_w['r2'],res_m['r2']],'MA
E':[res_w['mae'],res_m['mae']]}).to_csv('metricas_modelo.csv',index=False)
    print("Arquivos: semana_previsto.png, mes_previsto.png, previsao_semanal.csv,
previsao_mensal.csv, metricas_modelo.csv")

if __name__=="__main__": main()
```