

SPRINT 10.8.1.3.1.1.1.12

June 29, 2025

```
[6]: # -*- coding: utf-8 -*-
"""
Script de preprocessament per al dataset de comerç electrònic brasiler
Objectiu: Neteja, transforma i enriqueix les dades per a analisi i modelització
Autor: [Oriac Gimeno Lozano]
Data: [03/07/2025]
"""

# =====
# 1. IMPORTS I CONFIGURACIÓ INICIAL
# =====

import pandas as pd # Manipulació i analisi de dades estructurades (DataFrames)
import seaborn as sns # Visualització estadística de dades (gràfics atractius)
import matplotlib.pyplot as plt # Creació de visualitzacions i gràfics bàsics
import matplotlib as mpl # Configuració avançada de paràmetres de visualització
from matplotlib.lines import Line2D # Creació de línies personalitzades per a gràfics
import os # Interacció amb el sistema operatiu (gestió de fitxers i directoris)
import gc # Garbage Collector per a gestió de memòria (alliberament manual)
import numpy as np # Computació numèrica eficient amb arrays i matemàtiques
import holidays # Base de dades de festius nacionals i regionals
from datetime import datetime # Manipulació de dates i hores (objectes temporals)
import warnings # Gestió d'avars i alertes durant l'execució
import time # Mesura de temps d'execució i control de temps
import traceback # Captura i manipulació d'errors (traces d'execució)
from tqdm import tqdm # Barres de progrés per a bucles i iteracions llargues
import calendar # Funcions relacionades amb calendaris (mesos, dies setmana)
from matplotlib.ticker import FuncFormatter # Format personalitzat per eixos de gràfics
import matplotlib.dates as mdates # Eines específiques per a dates en visualitzacions
from matplotlib.dates import DateFormatter # Format específic per a representació de dates
from prophet import Prophet # Model de prediccio de sèries temporals de Facebook
```

```

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, median_absolute_error, max_error # Mètriques d'avaluació de models
from xgboost import XGBRegressor # Model d'aprenentatge automàtic basat en arbres (boosting)
from statsmodels.tsa.statespace.sarimax import SARIMAX # Model estadístic per a sèries temporals (SARIMA)
import random

# Desactivar warnings
warnings.filterwarnings('ignore', category=pd.errors.PerformanceWarning)
warnings.filterwarnings('ignore', category=FutureWarning)
warnings.filterwarnings('ignore', category=UserWarning)

# Configuració global
plt.style.use('ggplot')
mpl.rcParams['figure.figsize'] = (12, 6)
mpl.rcParams['axes.titlesize'] = 16
sns.set_palette("viridis")

rcParams = plt.rcParams
rcParams['figure.dpi'] = 300
rcParams['savefig.dpi'] = 300
rcParams['font.family'] = 'DejaVu Sans'
rcParams['axes.grid'] = True
rcParams['grid.alpha'] = 0.1

pd.set_option('display.max_columns', None)
pd.set_option('display.float_format', lambda x: '%.3f' % x)

# =====
# 2. DICCIÓNARI DE TRADUCCIONS I CONSTANTS
# =====

TRADUCCIONS_CA = {
    "health_beauty": "salut_i_bellesa",
    "computers_accessories": "accessoris_informàtics",
    "auto": "automòbil",
    "bed_bath_table": "llit_bany_taula",
    "furniture_decor": "mobles_decoració",
    "sports_leisure": "esports_oci",
    "perfumery": "perfumeria",
    "housewares": "articles_llar",
    "telephony": "telefonia",
    "watches_gifts": "rellotges_regals",
    "food_drink": "aliments_begudes",
    "baby": "nadons",
    "stationery": "papereria",
    "tablets_printing_image": "tauletes_impressió_imatge",
}

```

```

"toys": "joguines",
"fixed_telephony": "telefonia_fixa",
"garden_tools": "eines_jardí",
"fashion_bags_accessories": "moda_bolsos_accessoris",
"small_appliances": "electrodomèstics_petits",
"consoles_games": "consoles_jocs",
"audio": "àudio",
"fashion_shoes": "moda_calçat",
"cool_stuff": "productes_interessants",
"luggage_accessories": "equipatge_accessoris",
"air_conditioning": "aire_condicionat",
"construction_tools_construction": "eines_construcció",
"kitchen_dining_laundry_garden_furniture": "mobles_cuina_menjador_rentavanderia_jardí",
"construction_tools_garden": "eines_construcció_jardí",
"construction_tools_garden": "eines_construcció_jardí",
"fashion_male_clothing": "moda_roba_home",
"pet_shop": "botiga_mascotes",
"office_furniture": "mobles_oficina",
"market_place": "mercat",
"electronics": "electrònica",
"home_appliances": "electrodomèstics_llar",
"party_supplies": "subministres_festa",
"home_comfort": "llar_comfort",
"construction_tools_tools": "eines_construcció",
"construction_tools_tools": "eines_construcció",
"agro_industry_and_commerce": "agroindústria_comerç",
"furniture_mattress_and_upholstery": "mobles_matalàs_tapisseria",
"books_technical": "llibre_tècnics",
"home_construction": "construcció_llar",
"musical_instruments": "instruments_musicals",
"furniture_living_room": "mobles_sala_estar",
"construction_tools_lights": "eines_construcció_il·luminació",
"industry_commerce_and_business": "indústria_comerç_negocis",
"food": "aliments",
"art": "art",
"furniture_bedroom": "mobles_dormitori",
"books_general_interest": "llibres_interès_general",
"construction_tools_safety": "eines_construcció_seguretat",
"fashion_underwear_beach": "moda_roba_interior_platja",
"fashion_sport": "moda_esportiva",
"signaling_and_security": "senyalització_seguretat",
"computers": "ordinadors",
"christmas_supplies": "suministres_nadal",
"fashion_female_clothing": "moda_roba_dona",
"home_appliances_2": "electrodomèstics_llar",
"books_imported": "llibres_importats",

```

```

"drinks": "begudes",
"cine_photo": "cinema_fotografia",
"la_cuisine": "cuina",
"music": "música",
"home_comfort_2": "llar_confort",
"small_appliances_home_oven_and_coffee": "electrodomèstics_cocina",
"electrodomèstics_petits_forn_cafè",
"cds_dvds_musicals": "cds_dvds_musicals",
"dvds_blu_ray": "dvds_blu_ray",
"flowers": "flors",
"arts_and_craftmanship": "arts_artesania",
"diapers_and_hygiene": "bolquers_higiene",
"fashion_childrens_clothes": "moda_roba_infantil",
"security_and_services": "seguretat_serveis",
"fashio_female_clothing": "moda_roba_dona",
}

}BASE_PATH = r"G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT\u
↪10\Brazilian E-Commerce Public Dataset"

ARXIUS = {
    'orders': 'olist_orders_dataset.csv',
    'customers': 'olist_customers_dataset.csv',
    'order_items': 'olist_order_items_dataset.csv',
    'products': 'olist_products_dataset.csv',
    'traduccio': 'product_category_name_translation.csv',
    'reviews': 'olist_order_reviews_dataset.csv',
    'sellers': 'olist_sellers_dataset.csv',
    'payments': 'olist_order_payments_dataset.csv'
}

TIMESTAMP_COLS = {
    'orders': [
        'order_purchase_timestamp', 'order_approved_at',
        'order_delivered_carrier_date', 'order_delivered_customer_date',
        'order_estimated_delivery_date'
    ],
    'order_items': ['shipping_limit_date'],
    'reviews': ['review_creation_date', 'review_answer_timestamp']
}

MESOS_CATALA = {
    1: 'Gener', 2: 'Febrer', 3: 'Març', 4: 'Abril',
    5: 'Maig', 6: 'Juny', 7: 'Juliol', 8: 'Agost',
    9: 'Setembre', 10: 'Octubre', 11: 'Novembre', 12: 'Desembre'
}

```

```

DIES_SETMANA_CATALA = {
    'Monday': 'Dilluns',
    'Tuesday': 'Dimarts',
    'Wednesday': 'Dimecres',
    'Thursday': 'Dijous',
    'Friday': 'Divendres',
    'Saturday': 'Dissabte',
    'Sunday': 'Diumenge'
}

DIES_ORDRE = ['Dilluns', 'Dimarts', 'Dimecres', 'Dijous', 'Divendres', ↵
    ↵'Dissabte', 'Diumenge']

CATEGORIES_ESPECIALS = ['sense_traduccio', 'altres']

# =====
# 3. FUNCIONS AUXILIARS
# =====

def convertir_timestamps(df, nom_taula):
    if nom_taula in TIMESTAMP_COLS:
        cols = [col for col in TIMESTAMP_COLS[nom_taula] if col in df.columns]

        if cols:
            df[cols] = df[cols].apply(
                lambda col: pd.to_datetime(col, errors='coerce', ↵
                ↵format='%Y-%m-%d %H:%M:%S'))
    return df

def gestionar_valors_nuls_temporals(df):
    total_files = len(df)
    print(f" Total de registres a processar: {total_files},")

    if 'order_approved_at' in df.columns:
        null_count = df['order_approved_at'].isna().sum()
        if null_count > 0:
            df['order_approved_at'] = df['order_approved_at'].fillna(df['order_purchase_timestamp'])
            print(f" order_approved_at: {null_count} valors imputats amb ↵
            ↵order_purchase_timestamp")

    if 'order_delivered_carrier_date' in df.columns:
        mask = df['order_delivered_carrier_date'].notna()

        if mask.any():
            avg_process_time = (df.loc[mask, 'order_delivered_carrier_date'] - ↵
                df.loc[mask, 'order_purchase_timestamp']).mean()

```

```

        if not pd.isna(avg_process_time):
            null_mask = df['order_delivered_carrier_date'].isna()
            null_count = null_mask.sum()

            if null_count > 0:
                percent = (null_count / total_files) * 100
                df.loc[null_mask, 'order_delivered_carrier_date'] = (
                    df.loc[null_mask, 'order_purchase_timestamp'] +_
                    avg_process_time)
                print(f" order_delivered_carrier_date: {null_count}_
                    ({percent:.2f}%) valors imputats")
            else:
                print(" No s'ha pogut calcular temps mitjà per a_
                    order_delivered_carrier_date")

        if 'order_delivered_customer_date' in df.columns:
            mask = df['order_delivered_customer_date'].notna() &_
            df['order_delivered_carrier_date'].notna()

            if mask.any():
                avg_transit_time = (df.loc[mask, 'order_delivered_customer_date'] -_
                    df.loc[mask, 'order_delivered_carrier_date'])._
                    mean()

                if not pd.isna(avg_transit_time):
                    null_mask = df['order_delivered_customer_date'].isna()
                    null_count = null_mask.sum()

                    if null_count > 0:
                        percent = (null_count / total_files) * 100
                        df.loc[null_mask, 'order_delivered_customer_date'] = (
                            df.loc[null_mask, 'order_delivered_carrier_date'] +_
                            avg_transit_time)
                        print(f" order_delivered_customer_date: {null_count}_
                            ({percent:.2f}%) valors imputats")
                    else:
                        print(" No s'ha pogut calcular temps mitjà per a_
                            order_delivered_customer_date")

    return df

def optimitzar_memoria(df):
    total_files = len(df)

    if 'categoria_catala' in df.columns:
        counts = df['categoria_catala'].value_counts()

```

```

categories_freq = counts[counts > 100].index.tolist()

df['categoria_catala'] = df['categoria_catala'].apply(
    lambda x: x if x in categories_freq else 'altres')
df['categoria_catala'] = df['categoria_catala'].astype('category')
print(f" Categories catalanes optimitzades. Cardinalitat: {df['categoria_catala'].nunique()}")


cat_cols = ['seller_city', 'seller_state']
for col in cat_cols:
    if col in df.columns:
        if df[col].nunique() < 1000:
            df[col] = df[col].astype('category')


float_map = {
    'review_score': 'float32',
    'price': 'float32',
    'freight_value': 'float32'
}

for col, dtype in float_map.items():
    if col in df.columns:
        df[col] = df[col].astype(dtype)

if 'product_photos_qty' in df.columns:
    df['product_photos_qty'] = df.groupby('categoria_catala')['product_photos_qty'].transform(
        lambda x: x.fillna(x.mode()[0] if not x.mode().empty else 1))
    nan_after = df['product_photos_qty'].isna().sum()
    print(f" product_photos_qty: Valors NaN imputats. Nuls restants: {nan_after}")


int_cols = ['product_weight_g', 'product_length_cm', 'product_height_cm', 'product_width_cm']
for col in int_cols:
    if col in df.columns:
        nan_count = df[col].isna().sum()
        if nan_count > 0:
            percent = (nan_count / total_files) * 100
            df[col] = df[col].fillna(0)
            print(f" {col}: {nan_count} ({percent:.2f}%) valors NaN imputats a 0")
        df[col] = pd.to_numeric(df[col], errors='coerce', downcast='integer').astype('int32')


return df

```

```

def afegir_variables_temporals(df):
    if 'order_purchase_timestamp' not in df.columns:
        print(" No s'ha trobat la columna 'order_purchase_timestamp' ")
        return df

    print(" Afegint variables temporals...")

    timestamp = df['order_purchase_timestamp']

    # Variables bàsiques
    df['any'] = timestamp.dt.year.astype('int16')
    df['mes'] = timestamp.dt.month.astype('int8')
    df['dia_any'] = timestamp.dt.dayofyear.astype('int16')
    df['hora'] = timestamp.dt.hour.astype('int8')
    df['dia_setmana'] = timestamp.dt.dayofweek.astype('int8')
    df['dia_setmana_nom'] = timestamp.dt.day_name().astype('category')

    data_ordre = timestamp.dt.date

    # Festius nacionals de Brasil
    anys = df['any'].unique()
    br_holidays = holidays.Brazil(years=anyys.tolist())
    festius_set = set(br_holidays.keys())
    df['festiu_nacional'] = data_ordre.isin(festius_set).astype('int8')

    # Vacances d'estiu (Brasil)
    mes = timestamp.dt.month
    dia = timestamp.dt.day
    df['vacances_estiu'] = (
        ((mes == 12) & (dia >= 15)) |
        (mes.isin([1, 2])) |
        ((mes == 3) & (dia <= 15))
    ).astype('int8')

    # Carnaval (ja existent)
    df['carnaval'] = 0
    carnaval_dates = {
        2016: [pd.Timestamp('2016-02-05'), pd.Timestamp('2016-02-10')],
        2017: [pd.Timestamp('2017-02-24'), pd.Timestamp('2017-03-01')],
        2018: [pd.Timestamp('2018-02-09'), pd.Timestamp('2018-02-14')]
    }
    for anyo, dates in carnaval_dates.items():
        mascara_any = (timestamp >= dates[0]) & (timestamp <= dates[1])
        df.loc[mascara_any, 'carnaval'] = 1
    df['carnaval'] = df['carnaval'].astype('int8')

```

```

# NOUS: Esdeveniments comercials brasilers
df['dia_mares'] = 0
df['dia_pares'] = 0
df['dia_nens'] = 0
df['black_friday'] = 0
df['festas_juninas'] = 0

for anyo in anys:
    # Dia das Mäes (2n diumenge de maig)
    dia_mares = pd.Timestamp(f'{anyo}-05-01') + pd.DateOffset(weeks=1, □
    ↵weekday=6)
    df.loc[timestamp.dt.date == dia_mares.date(), 'dia_mares'] = 1

    # Dia dos Pais (2n diumenge d'agost)
    dia_pares = pd.Timestamp(f'{anyo}-08-01') + pd.DateOffset(weeks=1, □
    ↵weekday=6)
    df.loc[timestamp.dt.date == dia_pares.date(), 'dia_pares'] = 1

    # Dia das Crianças (12 d'octubre)
    df.loc[(timestamp.dt.month == 10) & (timestamp.dt.day == 12), □
    ↵'dia_nens'] = 1

    # Black Friday (4t dijous de novembre)
    novembre = pd.Timestamp(f'{anyo}-11-01')
    # Trobar el primer dijous
    primer_dijous = novembre + pd.DateOffset(days=(3 - novembre.dayofweek) □
    ↵% 7)
    black_friday = primer_dijous + pd.DateOffset(weeks=3, days=1) # 4t □
    ↵dijous + 1 dia
    df.loc[timestamp.dt.date == black_friday.date(), 'black_friday'] = 1

    # Festas Juninas (tot el mes de juny)
    df.loc[timestamp.dt.month == 6, 'festas_juninas'] = 1

# Variables existents
df['final_setmana'] = (df['dia_setmana'] >= 5).astype('int8')
df['trimestre'] = timestamp.dt.quarter.astype('int8')
df['setmana_mes'] = ((timestamp.dt.day - 1) // 7 + 1).astype('int8')
df['mes_nom'] = df['mes'].map(MESOS_CATALA).astype('category')

# Periode comercial (ja existent)
condicions = [
    (timestamp >= '2016-09-04') & (timestamp < '2017-09-04'),
    (timestamp >= '2017-09-04') & (timestamp < '2018-09-04'),
    (timestamp >= '2018-09-04')
]
opcions = ['2016-2017', '2017-2018', '2018-2019']

```

```

df['periode_comercial'] = np.select(condicions, opcions, u
↳default='fora_de_periode')
df['periode_comercial'] = df['periode_comercial'].astype('category')

# Resum de variables afegides
print(f" S'han afegit {len(['dia_mares', 'dia_pares', 'dia_nens', u
↳'black_friday', 'festas_juninas'])} noves variables comercials brasileres")
print(f" Variables temporals totals: {len(['any', 'mes', 'dia_any', u
↳'hora', 'dia_setmana', 'dia_setmana_nom', 'festiu_nacional', u
↳'vacances_estiu', 'carnaval', 'final_setmana', 'trimestre', 'setmana_mes', u
↳'mes_nom', 'periode_comercial', 'dia_mares', 'dia_pares', 'dia_nens', u
↳'black_friday', 'festas_juninas'])}")
#####
# NOU: Afegir dies de lliurament
if 'order_delivered_customer_date' in df.columns and u
↳'order_purchase_timestamp' in df.columns:
    # Assegurar que les dates són del tipus correcte
    if not pd.api.types.
        is_datetime64_any_dtype(df['order_delivered_customer_date']):
            df['order_delivered_customer_date'] = pd.
        to_datetime(df['order_delivered_customer_date'])

    if not pd.api.types.
        is_datetime64_any_dtype(df['order_purchase_timestamp']):
            df['order_purchase_timestamp'] = pd.
        to_datetime(df['order_purchase_timestamp'])

    # Calcular diferència en dies
df['delivery_days'] = (df['order_delivered_customer_date'] - u
↳df['order_purchase_timestamp']).dt.days

# Verificar valors negatius
if (df['delivery_days'] < 0).any():
    print(" S'han detectat valors negatius a delivery_days. Convertint u
↳a zero.")
    df['delivery_days'] = df['delivery_days'].clip(lower=0)

print(" Variable 'delivery_days' afegida correctament")

return df

def generar_informe_final(df, output_path):
    print("\n" + "="*50)
    print(" INFORME FINAL DE QUALITAT DE DADES")

```

```

print("\nIMPUTACIONS DE TEMPS:")
time_cols = ['order_approved_at', 'order_delivered_carrier_date', 'order_delivered_customer_date']
for col in time_cols:
    if col in df.columns:
        null_initial = df[col].isna().sum()
        if null_initial > 0:
            print(f"- {col}: Valors nuls inicials = {null_initial}")

print("\nIMPUTACIONS DE PRODUCTES:")
product_cols = ['product_photos_qty', 'product_weight_g',
                'product_length_cm', 'product_height_cm', 'product_width_cm']
for col in product_cols:
    if col in df.columns:
        zeros = (df[col] == 0).sum()
        if zeros > 0:
            percent = (zeros / len(df)) * 100
            print(f"- {col}: {zeros} ({percent:.2f}%) valors imputats a 0")

print("\nRESUM DE VARIABLES CLAU:")
if 'categoria_catala' in df.columns:
    print(f"- Categories de productes: {df['categoria_catala'].nunique()}")
    categories = df['categoria_catala'].nunique()
print(f"- Venedors únics: {df['seller_id'].nunique()}")
print(f"- Clients únics: {df['customer_id'].nunique()}")
print(f"- Interval temporal: De {df['order_purchase_timestamp'].min()} a "
      f"{df['order_purchase_timestamp'].max()}")


print("\nVARIABLES D'ESTACIONALITAT:")
print(f"- Dies festius: {df['festiu_nacional'].sum()} comandes "
      f"({df['festiu_nacional'].mean()*100:.1f}%)")
print(f"- Vacances d'estiu: {df['vacances_estiu'].sum()} comandes "
      f"({df['vacances_estiu'].mean()*100:.1f}%)")
print(f"- Carnaval: {df['carnaval'].sum()} comandes ({df['carnaval']. "
      f"mean()*100:.1f}%)")

if 'categoria_catala' in df.columns:
    print("\nDISTRIBUCIÓ DE CATEGORIES CATALANES:")
    top_categories = df['categoria_catala'].value_counts().head(10)
    for cat, count in top_categories.items():
        percent = count / len(df) * 100
        print(f"- {cat}: {count} comandes ({percent:.1f}%)")

print("\n" + "="*50)
print(" PROCÉS DE PREPROCESSAMENT COMPLETAT AMB ÒUT")

```

```

def verificar_relacions_i_qualitat(df_orders, df_payments, dfs_dict):
    print("\n" + "*50)
    print(" VERIFICACIÓ DE RELACIONS 1:N I QUALITAT DE DADES")

    try:
        assert df_orders['order_id'].nunique() == len(df_orders)
        print(" Relació 1:1 verificada: Ordres úniques")

        pagaments_per_ordre = df_payments.groupby('order_id').size()
        assert pagaments_per_ordre.max() == pagaments_per_ordre.min()
        print(" Relació 1:1 verificada: Pagaments únics per ordre")
    except AssertionError:
        print(" Avis: S'han detectat relacions 1:N en pagaments per ordre")
        print(f" - Màxim pagaments per ordre: {pagaments_per_ordre.max()}")
        print(f" - Mínim pagaments per ordre: {pagaments_per_ordre.min()}")
        print(f" - Mitjana pagaments per ordre: {pagaments_per_ordre.mean():.2f}")

def data_quality_report(df):
    report = pd.DataFrame({
        'missing_%': df.isnull().mean() * 100,
        'dtype': df.dtypes,
        'unique_values': df.nunique(),
        'exemple_valor': df.iloc[0] if len(df) > 0 else None
    })
    return report.sort_values('missing_%', ascending=False)

print("\n INFORME DE QUALITAT PER TAULA:")
for nom, df in dfs_dict.items():
    print(f"\n TAULA: {nom.upper()}")
    display(data_quality_report(df))

print("*50)

def afegir_normalitzacio_preus(df):
    print(" Normalitzant preus per categories...")

    epsilon = 1e-9

    df['price_normalized'] = df.groupby('categoria_catala')['price'].transform(
        lambda x: (x - x.min()) / (x.max() - x.min() + epsilon))

    print(f" Preus normalitzats per a {df['categoria_catala'].nunique()} categories")
    return df

```

```

def plot_facturacio_mensual(df):
    required_cols = ['order_purchase_timestamp', 'price']
    missing_cols = [col for col in required_cols if col not in df.columns]
    if missing_cols:
        raise ValueError(f" Falten columnes necessàries: {', '.join(missing_cols)}")

    df = df.copy()
    df['any_mes'] = df['order_purchase_timestamp'].dt.to_period('M').dt.
    ↪to_timestamp()
    facturacio = df.groupby('any_mes', as_index=False)['price'].sum()
    facturacio.rename(columns={'price': 'facturacio'}, inplace=True)

    plt.figure(figsize=(14, 7))
    bars = plt.bar(facturacio['any_mes'], facturacio['facturacio'] / 1000,
                   width=25, color='#3498db', alpha=0.7,
                   label='Facturació Mensual')

    # Calcular tendència si hi ha suficients dades
    pendent = None
    if len(facturacio) > 1:
        x = mdates.date2num(facturacio['any_mes'])
        y = facturacio['facturacio'] / 1000
        z = np.polyfit(x, y, 1)
        p = np.poly1d(z)
        plt.plot(facturacio['any_mes'], p(x),
                  linewidth=2.5, color="#e74c3c",
                  label="Tendència")
        pendent = z[0] # Guardar pendent per conclusions

    plt.title('Facturació Mensual i Tendència - Ecommerce Brasil', fontsize=16)
    plt.ylabel('Facturació (Milers BRL)', fontsize=12)
    plt.xlabel('Període', fontsize=12)
    plt.grid(True, linestyle='--', alpha=0.3, axis='y')

    ax = plt.gca()
    ax.xaxis.set_major_formatter(DateFormatter("%b '%y"))
    ax.xaxis.set_major_locator(mdates.MonthLocator(interval=2))
    plt.xticks(rotation=45)

    # Variables per conclusions
    max_mes_obj = None
    max_fact = 0

    # Identificar màxim
    if not facturacio.empty:
        max_idx = facturacio['facturacio'].idxmax()

```

```

max_mes_obj = facturacio.loc[max_idx, 'any_mes'] # Guardar objecte de l'estructura de la data
max_fact = facturacio.loc[max_idx, 'facturacio'] / 1000
plt.annotate(f" Màxim: BRL{max_fact:.0f}K",
             xy=(max_mes_obj, max_fact),
             xytext=(0, 10),
             textcoords='offset points',
             ha='center',
             fontsize=10,
             color='red')

plt.legend(loc='upper left')
for bar in bars:
    height = bar.get_height()
    if height > 0:
        plt.text(bar.get_x() + bar.get_width()/2., height,
                  f'BRL{height:.0f}K',
                  ha='center', va='bottom', fontsize=8, rotation=90)

plt.tight_layout()
plt.show()

# =====
# CONCLUSIONS NUMÈRIQUES
# =====
print("\n" + "="*50)
print(" CONCLUSIONS DE FACTURACIÓ MENSUAL")

if facturacio.empty:
    print(" No hi ha dades per mostrar conclusions")
    print("=="*50 + "\n")
    return facturacio

# 1. Període analitzat
min_mes = facturacio['any_mes'].min().strftime("%b '%y")
max_mes = facturacio['any_mes'].max().strftime("%b '%y")
print(f"• Període analitzat: {min_mes} a {max_mes}")

# 2. Facturació total
fact_total = facturacio['facturacio'].sum() / 1000 # En milers BRL
print(f"• Facturació total: BRL{fact_total:.0f}K")

# 3. Mesos amb màxim i mínim
min_idx = facturacio['facturacio'].idxmin()
min_mes_date = facturacio.loc[min_idx, 'any_mes']
min_val = facturacio.loc[min_idx, 'facturacio'] / 1000

```

```

print(f"• Mes amb mínima facturació: {min_mes_date.strftime('%b %y')}") ↵
(BRL[min_val:.0f}K)")

# Utilitzar max_mes_obj que està guardat
if max_mes_obj is not None:
    print(f"• Mes amb màxima facturació: {max_mes_obj.strftime('%b %y')}") ↵
(BRL[max_fact:.0f}K)")

# 4. Variació i creixement
primer_val = facturacio.iloc[0]['facturacio'] / 1000
darrer_val = facturacio.iloc[-1]['facturacio'] / 1000

if primer_val > 0: # Evitar divisió per zero
    variacio = ((darrer_val - primer_val) / primer_val) * 100
    print(f"• Creixement del període: {variacio:.1f}%")

# 5. Tendència (si existeix)
if pendent is not None:
    tendencia = "creixement" if pendent > 0 else "decreixement"
    print(f"• Tendència mensual: {abs(pendent):.1f}K BRL/mes ({tendencia})")

# 6. Recomanació basada en dades
if max_mes_obj:
    mes_maxim = max_mes_obj.month

    if mes_maxim == 11:
        print("\n RECOMANACIÓ: El pic de vendes coincideix amb el Black Friday.")
        print(" - Optimitzi campanyes de marketing per a aquest període.")
        print(" - Asseguri stock suficient per a la demanda.")
    elif mes_maxim == 12:
        print("\n RECOMANACIÓ: El màxim de vendes és en temporada nadalenca.")
        print(" - Consideri promocions d'última oportunitat.")

    print("-" * 50 + "\n")

return facturacio

def plot_heatmap_vendes_complet(df, output_dir=None):
    print(" Generant mapa de calor de vendes per trams horaris (periode complet) ...")

    if output_dir is None:
        output_dir = os.path.join(BASE_PATH, "visualitzacions")
    os.makedirs(output_dir, exist_ok=True)

```

```

df = df.copy()
df['price'] = pd.to_numeric(df['price'], errors='coerce')
df['hora'] = df['order_purchase_timestamp'].dt.hour
df['dia_cat'] = df['order_purchase_timestamp'].dt.day_name().
↪map(DIES_SETMANA_CATALA)

bins = list(range(0, 25, 2))
labels = [f"{i}:02d}:00-{i+2}:02d}:00" for i in range(0, 24, 2)]
df['tram_hores'] = pd.cut(df['hora'], bins=bins, labels=labels, right=False)

min_date = df['order_purchase_timestamp'].min().strftime('%Y-%m-%d')
max_date = df['order_purchase_timestamp'].max().strftime('%Y-%m-%d')

heatmap_data = (
    df.groupby(['tram_hores', 'dia_cat'], observed=True)['price']
    .sum()
    .unstack(fill_value=0)
    .reindex(columns=DIES_ORDRE, index=labels, fill_value=0)
    / 1000
)

if heatmap_data.sum().sum() == 0:
    print(" Avis: No s'han trobat dades de vendes")
    return None

plt.figure(figsize=(14, 8))
ax = sns.heatmap(
    heatmap_data,
    cmap="YlGnBu",
    annot=True,
    fmt=".1f",
    linewidths=0.5,
    cbar_kws={'label': 'Vendes (Milers BRL)'}
)

ax.set_title(f'Vendes per Trams de 2 Hores i Dia\nPeríode complet: {min_date} a {max_date}', fontsize=16)
ax.set_xlabel('Dia de la Setmana', fontsize=12)
ax.set_ylabel('Tram Horari', fontsize=12)
ax.set_xticklabels(DIES_ORDRE, rotation=45, ha='right')

plt.tight_layout()
filename = "heatmap_vendes_periode_complet.png"
output_path = os.path.join(output_dir, filename)
plt.savefig(output_path, dpi=300, bbox_inches='tight')
print(f" Mapa de calor guardat a: {output_path}")
plt.show()

```

```

    return heatmap_data

def plot_top10_periode(df, periode, output_dir=None):
    df_periode = df[df['periode_comercial'] == periode]
    facturacio = df_periode.groupby('categoria_catala')['price'].sum() .
    ↪reset_index()
    facturacio = facturacio.sort_values('price', ascending=False).head(10)
    facturacio = facturacio.sort_values('price', ascending=True)

    plt.figure(figsize=(12, 8))
    plt.barh(
        facturacio['categoria_catala'],
        facturacio['price'] / 1000,
        color='#3498db',
        alpha=0.7
    )

    for i, valor in enumerate(facturacio['price'] / 1000):
        plt.text(
            valor,
            i,
            f'BRL{valor:.0f}K',
            va='center',
            fontsize=10,
            fontweight='bold'
        )

    plt.title(f'Top 10 Categories per Facturació ({periode})', fontsize=18)
    plt.xlabel('Facturació (Milers BRL)', fontsize=14)
    plt.ylabel('Categoria', fontsize=14)
    plt.grid(axis='x', alpha=0.3)

    plt.tight_layout()

    if output_dir:
        os.makedirs(output_dir, exist_ok=True)
        filename = f"top10_categories_{periode.replace('-', '_)}.png"
        output_path = os.path.join(output_dir, filename)
        plt.savefig(output_path, dpi=300, bbox_inches='tight')
        print(f" Gràfic guardat a: {output_path}")

    plt.show()
    return facturacio

def plot_top10_periode_2016_2017(df, output_dir=None):
    return plot_top10_periode(df, '2016-2017', output_dir)

```

```

def plot_top10_periode_2017_2018(df, output_dir=None):
    return plot_top10_periode(df, '2017-2018', output_dir)
def plot_comparativa_facturacio_periodes(top_2016_2017, top_2017_2018,□
    ↵output_dir):
    """Compara la facturació de les mateixes categories en dos períodes□
    ↵diferents"""
    plt.figure(figsize=(14, 8))

    # Fusionar les dades dels dos períodes
    df_comparativa = pd.merge(
        top_2016_2017,
        top_2017_2018,
        on='categoria_catala',
        suffixes=('_2016', '_2017')
    )

    # Preparar dades per al gràfic de barres agrupades
    categories = df_comparativa['categoria_catala']
    fact_2016 = df_comparativa['price_2016'] / 1000
    fact_2017 = df_comparativa['price_2017'] / 1000

    x = np.arange(len(categories))
    width = 0.35

    plt.bar(x - width/2, fact_2016, width, label='2016-2017', color="#3498db")
    plt.bar(x + width/2, fact_2017, width, label='2017-2018', color="#e74c3c")

    plt.title('Comparativa de Facturació per Categoria entre Períodes',□
    ↵fontsize=18)
    plt.ylabel('Facturació (Milers BRL)', fontsize=14)
    plt.xlabel('Categories', fontsize=14)
    plt.xticks(x, categories, rotation=45, ha='right')
    plt.legend()
    plt.grid(axis='y', alpha=0.3)

    # Afegir valors a les barres
    for i, val in enumerate(fact_2016):
        plt.text(i - width/2, val + 5, f'BRL{val:.0f}K', ha='center',□
        ↵fontsize=9)
    for i, val in enumerate(fact_2017):
        plt.text(i + width/2, val + 5, f'BRL{val:.0f}K', ha='center',□
        ↵fontsize=9)

    plt.tight_layout()
    output_path = os.path.join(output_dir, "comparativa_facturacio_periodes.□
    ↵png")

```

```

plt.savefig(output_path, dpi=300, bbox_inches='tight')
plt.show()
print(f" Comparativa de facturació guardada a: {output_path}")

def plot_top10_periode_units(df, periode, output_dir=None):
    df_periode = df[df['periode_comercial'] == periode]
    unitats = df_periode.groupby('categoria_catala')['order_item_id'].count().
    ↵reset_index()
    unitats = unitats.rename(columns={'order_item_id': 'unitats_venudes'})
    unitats = unitats.sort_values('unitats_venudes', ascending=False).head(10)
    unitats = unitats.sort_values('unitats_venudes', ascending=True)

    plt.figure(figsize=(12, 8))
    plt.barh(unitats['categoria_catala'], unitats['unitats_venudes'], color="#2ecc71", alpha=0.7)

    for i, valor in enumerate(unitats['unitats_venudes']):
        plt.text(valor, i, f'{valor:,}', va='center', fontsize=10, fontweight='bold')

    plt.title(f'Top 10 Categories per Unitats Venudes ({periode})', fontsize=18)
    plt.xlabel('Unitats Venudes', fontsize=14)
    plt.ylabel('Categoria', fontsize=14)
    plt.grid(axis='x', alpha=0.3)

    if output_dir:
        os.makedirs(output_dir, exist_ok=True)
        filename = f"top10_categories_units_{periode.replace('-', '_)}.png"
        plt.savefig(os.path.join(output_dir, filename), dpi=300, bbox_inches='tight')

    plt.show()
    return unitats

def plot_top10_periode_2016_2017_units(df, output_dir=None):
    return plot_top10_periode_units(df, '2016-2017', output_dir)

def plot_top10_periode_2017_2018_units(df, output_dir=None):
    return plot_top10_periode_units(df, '2017-2018', output_dir)

def plot_comparativa_unitats_periodes(top_units_2016_2017, top_units_2017_2018, ↵
    ↵output_dir):
    """Compara les unitats venudes de les mateixes categories en dos períodes diferents"""
    plt.figure(figsize=(14, 8))

```

```

# Fusionar les dades dels dos períodes
df_comparativa = pd.merge(
    top_units_2016_2017,
    top_units_2017_2018,
    on='categoria_catala',
    suffixes=('_2016', '_2017')
)

# Preparar dades per al gràfic de barres agrupades
categories = df_comparativa['categoria_catala']
units_2016 = df_comparativa['unitats_venudes_2016']
units_2017 = df_comparativa['unitats_venudes_2017']

x = np.arange(len(categories))
width = 0.35

plt.bar(x - width/2, units_2016, width, label='2016-2017', color="#2ecc71")
plt.bar(x + width/2, units_2017, width, label='2017-2018', color="#9b59b6")

plt.title('Comparativa d\'Unitats Venudes per Categoria entre Períodes', fontsize=18)
plt.ylabel('Unitats Venudes', fontsize=14)
plt.xlabel('Categories', fontsize=14)
plt.xticks(x, categories, rotation=45, ha='right')
plt.legend()
plt.grid(axis='y', alpha=0.3)

# Afegir valors a les barres
for i, val in enumerate(units_2016):
    plt.text(i - width/2, val + 10, f'{val:,}', ha='center', fontsize=9)
for i, val in enumerate(units_2017):
    plt.text(i + width/2, val + 10, f'{val:,}', ha='center', fontsize=9)

plt.tight_layout()
output_path = os.path.join(output_dir, "comparativa_unitats_períodes.png")
plt.savefig(output_path, dpi=300, bbox_inches='tight')
plt.show()
print(f" Comparativa d'unitats guardada a: {output_path}")

def matriu_correlacio_millorada(df, visualitzacions_dir, cols, filename):
    """Versió robusta amb gestió d'errors millorada"""
    try:
        # 1. Verificar columnes
        missing = [col for col in cols if col not in df.columns]
        if missing:
            print(f" ERROR: Columnes faltants: {missing}")
            return None

```

```

# 2. Preparar dades
df_corr = df[cols].copy().dropna()

if len(df_corr) < 10:
    print(f" AVÍS: Només {len(df_corr)} registres vàlids -_
→correlacions poc fiables")

# 3. Calcular correlacions
try:
    corr_spearman = df_corr.corr(method='spearman')
    corr_kendall = df_corr.corr(method='kendall')
except Exception as e:
    print(f" ERROR calculant correlacions: {str(e)}")
    return None

# 4. Crear matriu combinada
corr_combined = corr_spearman.copy().astype(object)
for i in range(len(corr_spearman)):
    for j in range(len(corr_spearman)):
        spear_val = corr_spearman.iloc[i, j]
        kendall_val = corr_kendall.iloc[i, j]
        corr_combined.iloc[i, j] = f"{spear_val:.2f}\n{kendall_val:.
→2f})"

# 5. Configurar gràfic
plt.figure(figsize=(12, 10))
ax = sns.heatmap(
    corr_spearman,
    annot=corr_combined.values,
    fmt="",
    cmap='coolwarm',
    vmin=-1,
    vmax=1,
    linewidths=0.8,
    linecolor='gray',
    cbar_kws={'label': 'Intensitat de Correlació', 'shrink': 0.8},
    annot_kws={
        "size": 12,
        "va": 'center',
        "bbox": dict(boxstyle="round,pad=0.3", fc="white", ec="gray",_
→alpha=0.7)
    }
)

# 6. Afegir títol i etiquetes

```

```

plt.title('Matriu de Correlació Combinada\nSpearman (superior) i\nKendall (inferior)', fontsize=18, pad=20)
plt.xticks(fontsize=12, rotation=45, ha='right')
plt.yticks(fontsize=12, rotation=0)

# 7. Afegir quadre informatiu
text_box = "Spearman: Relacions monòtones\nKendall: Concordància de\nparells"
plt.figtext(0.5, -0.05, text_box, ha="center", fontsize=12, color="orange", alpha=0.2, bbox={"facecolor": "white", "alpha": 0.2, "pad": 5})

# 8. Guardar i mostrar
plt.tight_layout()
output_path = os.path.join(visualitzacions_dir, filename)
plt.savefig(output_path, dpi=300, bbox_inches='tight')
print(f" Matriu de correlació guardada a: {output_path}")
plt.show()

# 9. Anàlisi de conclusions
print("\n CONCLUSIONS DE CORRELACIÓ:")

# Diccionari de noms descriptius
nom_variables = {
    'review_score': 'Satisfacció del client',
    'delivery_days': 'Dies de lliurament',
    'price_normalized': 'Preu normalitzat'
}

# Analitzar cada parella de variables
for i, var1 in enumerate(cols):
    for j, var2 in enumerate(cols):
        if i < j: # Evitar duplicats i auto-correlacions
            spear = corr_spearman.iloc[i, j]
            kendall = corr_kendall.iloc[i, j]

            nom1 = nom_variables.get(var1, var1)
            nom2 = nom_variables.get(var2, var2)

            print(f"\n Relació entre {nom1} i {nom2}:")
            print(f" - Spearman: {spear:.3f}, Kendall: {kendall:.3f}")

# Interpretar força i direcció
for metode, valor in [('Spearman', spear), ('Kendall', kendall)]:
    if abs(valor) > 0.7:
        força = "molt forta"
    elif abs(valor) > 0.5:
        força = "forta"
    else:
        força = "modera"
    print(f" La relació entre {nom1} i {nom2} és {força} i {metode}. El valor de la correlació {metode} és {valor:.3f}.")
```

```

        força = "força significativa"
    elif abs(valor) > 0.3:
        força = "moderada"
    elif abs(valor) > 0.1:
        força = "feble"
    else:
        força = "pràcticament nul·la"

    if valor > 0:
        direcció = "positiva"
        interpretació = f"quan {nom1} augmenta, {nom2} ↴
        ↵tendeix a augmentar"
    elif valor < 0:
        direcció = "negativa"
        interpretació = f"quan {nom1} augmenta, {nom2} ↴
        ↵tendeix a disminuir"
    else:
        direcció = "sense direcció clara"
        interpretació = "no hi ha relació aparent"

    print(f" - {metode}: Correlació {força} i {direcció} ↴
        ↵({interpretació})")

# Interpretació específica per a relacions clau
if var1 == 'review_score' and var2 == 'delivery_days':
    if spear < -0.3 or kendall < -0.3:
        print("      INSIGHT IMPORTANT: Hi ha evidència que ↴
        ↵els temps de lliurament més llargs "
              "estan associats amb menor satisfacció dels ↴
        ↵clients.")
    elif spear > 0.1 or kendall > 0.1:
        print("      RESULTAT INESPERAT: S'observa una ↴
        ↵relació positiva inesperada entre "
              "temps de lliurament i satisfacció. Cal ↴
        ↵investigar possibles causes.")

if var1 == 'review_score' and var2 == 'price_normalized':
    if spear < -0.3 or kendall < -0.3:
        print("      INSIGHT IMPORTANT: Els preus més alts ↴
        ↵estan associats amb menor satisfacció,
              "potser per expectatives més altes no ↴
        ↵satisfetes.")
    elif spear > 0.3 or kendall > 0.3:
        print("      INSIGHT IMPORTANT: Els preus més alts ↴
        ↵estan associats amb major satisfacció, "

```

```

        "suggerint que els clients percepren millor la
        ↪qualitat en productes cars.")

    return {
        'spearman': corr_spearman,
        'kendall': corr_kendall
    }

except Exception as e:
    print(f"  ERROR NO CONTROLAT a matriu_correlacio_millorada: {str(e)}")
    return None

def plot_estacionalitat_anual_per_categoria(df):
    print("\n  Generant estacionalitat anual per categoria...")

    output_dir = os.path.join(BASE_PATH, "visualitzacions")
    os.makedirs(output_dir, exist_ok=True)
    start_time = time.time()

    df = df.copy()
    df['any'] = df['order_purchase_timestamp'].dt.year
    df['mes'] = df['order_purchase_timestamp'].dt.month

    anys_complets = df.groupby('any')['mes'].nunique()
    anys_disponibles = anys_complets.index.tolist()
    print(f"  Anys disponibles: {anydisponibles}")

    palette_anys = {
        2016: '#1f77b4',
        2017: '#ff7f0e',
        2018: '#2ca02c'
    }

    categories = df['categoria_catala'].unique()
    n_categories = len(categories)
    n_cols = 4
    n_rows = int(np.ceil(n_categories / n_cols))

    fig, axs = plt.subplots(
        n_rows, n_cols,
        figsize=(22, n_rows*5),
        sharex=True,
        sharey=True
    )

    fig.suptitle('Estacionalitat vendes anual per categoria (2016-2018)',
                 fontsize=24, fontweight='bold', y=0.98)

```

```

    axs_flat = axs.flatten() if n_categories > 1 else [axs]

    for i, categoria in enumerate(categories):
        ax = axs_flat[i]
        df_categoria = df[df['categoria_catala'] == categoria]

        for anyo in sorted(anys_disponibles):
            df_any = df_categoria[df_categoria['any'] == anyo]
            df_any_agregat = df_any.groupby('mes')[['price']].sum().reset_index()

            if not df_any_agregat.empty:
                min_val = df_any_agregat['price'].min()
                max_val = df_any_agregat['price'].max()
                rang = max_val - min_val
                df_any_agregat['vendes_normalitzades'] = ↵
                    (df_any_agregat['price'] - min_val) / (rang + 1e-9)

                sns.lineplot(
                    x='mes',
                    y='vendes_normalitzades',
                    data=df_any_agregat,
                    ax=ax,
                    marker='o',
                    markersize=6,
                    linewidth=2.5,
                    color=palette_anys.get(anyo, '#777777'),
                    label=str(anyo))

            ax.set_title(categoria, fontsize=16, fontweight='bold', pad=12)
            ax.set_xlabel('')
            ax.set_ylabel('')
            ax.set_xticks(range(1, 13))
            ax.set_xticklabels([MESOS_CATALA[m] [:3] for m in range(1, 13)], ↵
                rotation=45, fontsize=12)
            if anys_disponibles:
                ax.legend(title='Any', title_fontsize=11, fontsize=10, loc='upper ↵
                left')
                ax.grid(True, linestyle='--', alpha=0.3)
                ax.axvline(x=9, color='red', linestyle='-', alpha=0.3, linewidth=3)

            for j in range(i+1, len(axs_flat)):
                axs_flat[j].axis('off')

    fig.text(0.5, 0.01, 'Mes', ha='center', fontsize=18)
    fig.text(0.01, 0.5, 'Vendes Normalitzades (0-1 per any)', va='center', ↵
        rotation='vertical', fontsize=18)
    fig.text(0.5, 0.001,

```

```

    "Nota: Les vendes estan normalitzades dins de cada any (0 = mínim d'any, 1 = màxim d'any) per comparar patrons d'estacionalitat.",
    ha='center', fontsize=12, style='italic')
plt.tight_layout(rect=[0.01, 0.01, 1, 0.96])
output_path = os.path.join(output_dir, "estacionalitat_anual_per_categoria.png")
plt.savefig(output_path, dpi=300, bbox_inches='tight')
print(f" Gràfic guardat a: {output_path}")
print(f" Temps de generació: {time.time() - start_time:.2f} segons")
plt.show()

return df

def analitzar_estacionalitat_per_categoria(df_final):
    if 'categoria_catala' not in df_final.columns or 'mes' not in df_final.columns:
        print(" Dades insuficients per a l'anàlisi d'estacionalitat")
        return None

    print("\n Realitzant anàlisi d'estacionalitat per categoria...")

    df_agregat = df_final.groupby(['categoria_catala', 'periode_comercial',
                                    'mes']).agg(
        vendes=('order_id', 'nunique'),
        ingressos=('price', 'sum'),
        satisfacció_mitjana=('review_score', 'mean'))
    .reset_index()

    df_agregat['mes_nom'] = df_agregat['mes'].map(MESOS_CATALA)
    df_agregat['creixement'] = df_agregat.groupby('categoria_catala')['vendes'].pct_change() * 100

    print(" Anàlisi d'estacionalitat completat")
    return df_agregat

def visualitzar_estacionalitat(df_agregat, categoria=None):
    if df_agregat is None or df_agregat.empty:
        print(" No hi ha dades per visualitzar")
        return

    output_dir = os.path.join(BASE_PATH, "visualitzacions")
    os.makedirs(output_dir, exist_ok=True)

    plt.style.use('ggplot')
    mpl.rcParams['figure.figsize'] = (14, 8)
    mpl.rcParams['axes.titlesize'] = 18
    mpl.rcParams['axes.labelsize'] = 14

```

```

mpl.rcParams['xtick.labelsize'] = 12
mpl.rcParams['ytick.labelsize'] = 12
mpl.rcParams['legend.fontsize'] = 12

# Filtrar períodes incomplets
if 'periode_comercial' in df_agregat.columns:
    periodes_complets = ['2016-2017', '2017-2018', '2018-2019', '2019-post']
    df_agregat = df_agregat[df_agregat['periode_comercial'].isin(periodes_complets)]

if categoria:
    # Visualització per categoria específica
    df_filt = df_agregat[df_agregat['categoria_catala'] == categoria]

    if df_filt.empty:
        print(f" No hi ha dades per a la categoria '{categoria}'")
        return

    # Obtenim els períodes presents (CORRECIÓ CLAU)
    periodes_presents = df_filt['periode_comercial'].unique()

    # Paleta universal
    full_palette = {
        '2016-2017': '#1f77b4',
        '2017-2018': '#ff7f0e',
        '2018-2019': '#2ca02c',
        '2019-post': '#d62728',
        'fora_de_periode': '#7f7f7f'
    }

    # Crear paleta només per als períodes presents
    palette = {}
    for periode in periodes_presents:
        if periode in full_palette:
            palette[periode] = full_palette[periode]
        else:
            # Generar color hex aleatori si falta
            palette[periode] = "#{:06x}".format(random.randint(0, 0xFFFFFF))
            print(f" Assignat color aleatori a període no previst:{periode}")
    plt.figure(figsize=(14, 8))

    sns.lineplot(
        data=df_filt,
        x='mes',
        y='vendes',

```

```

        hue='periode_comercial',
        style='periode_comercial',
        markers=True,
        dashes=False,
        markersize=10,
        linewidth=2.5,
        palette=palette
    )

plt.xticks(range(1, 13), [MESOS_CATALA[m] for m in range(1, 13)])
plt.xlim(0.5, 12.5)
plt.title(f"Estacionalitat de vendes: {categoria}", fontsize=18)
plt.xlabel('Mes', fontsize=14)
plt.ylabel('Nombre de Vendes', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7)

# Marcar inici de periode comercial (Setembre)
plt.axvline(x=9, color='gray', linestyle='--', alpha=0.7)
plt.text(9.1, plt.ylim()[1]*0.95, 'Inici Període',
         rotation=90, verticalalignment='top', fontsize=12)

plt.legend(title='Període Comercial', title_fontsize=13,
           loc='best', frameon=True, framealpha=0.9)

plt.figtext(0.5, 0.01,
            "Període 2016-2017: 04/09/2016 - 03/09/2017 | "
            "Període 2017-2018: 04/09/2017 - 03/09/2018",
            ha="center", fontsize=11, bbox={"facecolor": "white", "alpha": 0.7, "pad": 5})

filename = f"estacionalitat_{categoria}.png"
output_path = os.path.join(output_dir, filename)
plt.tight_layout()
plt.savefig(output_path, dpi=300, bbox_inches='tight')
print(f" Gràfic guardat a: {output_path}")
plt.show()

else:
    # Crear heatmap general
    heatmap_data = df_agregat.groupby(['categoria_catala', 'mes'])['vendes'].mean().unstack()

    # Normalització de 0 a 1 per categoria (respecte al màxim)
    heatmap_data = heatmap_data.div(heatmap_data.max(axis=1), axis=0)

```

```

# Ordenar categories per vendes totals
category_order = df_agregat.groupby('categoria_catala')['vendes'].sum() .
sort_values(ascending=False).index
heatmap_data = heatmap_data.loc[category_order]

# Configuració del heatmap
plt.figure(figsize=(16, 14)) # Més alt per a les etiquetes dels events
ax = sns.heatmap(
    heatmap_data,
    cmap="YlGnBu",
    annot=True,
    fmt=".1f",
    linewidths=.5,
    annot_kws={"size": 9},
    cbar_kws={'label': 'Vendes Normalitzades (0=min, 1=max)'}
)

plt.title('Mapa de Calor: Estacionalitat per Categoria', fontsize=18, u
pad=20)
plt.xlabel('Mes', fontsize=14)
plt.ylabel('Categoria', fontsize=14)
ax.set_xticklabels([MESOS_CATALA[i] for i in range(1, 13)], rotation=45)

# Marcar inici de període comercial (Setembre)
plt.axvline(x=8.5, color='blue', linestyle='-', alpha=0.5, linewidth=2)
plt.text(8.6, len(heatmap_data) + 0.5, 'Inici Període Comercial',
        rotation=90, verticalalignment='top', fontsize=12, color='blue')

# Llegenda d'esdeveniments brasilers (sense sobreposició)
events = {
    1: "Reis/Volta às Aulas",
    2: "Carnaval",
    4: "Páscoa",
    5: "Dia das Mães",
    6: "Festas Juninas",
    8: "Dia dos Pais",
    10: "Dia das Crianças",
    11: "Black Friday",
    12: "Natal/Réveillon"
}

# Afegir línies verticals per a events
for mes in events.keys():
    plt.axvline(x=mes-0.5, color='red', linestyle='-', alpha=0.3, u
linewidth=1)

# Crear llegenda separada per a events

```

```

#legend_elements = [Line2D([0], [0], color='red', lw=2, alpha=0.3, u
˓→label='Esdeveniment Brasiler')]
plt.legend(handles=legend_elements, loc='upper center',
#           bbox_to_anchor=(0.5, -0.05), ncol=3, frameon=True, u
˓→framealpha=0.8)

# Afegir taula d'esdeveniments al peu del gràfic
event_text = " | ".join([f"{MESOS_CATALA[mes]}: {event}" for mes, event u
˓→in events.items()])
plt.figtext(0.5, 0.01, "ESDEVENIMENTS: " + event_text,
            ha="center", fontsize=10, bbox={"facecolor": "white", "alpha": :
˓→0.7, "pad": 5})

# Guardar i mostrar
filename = "estacionalitat_general.png"
output_path = os.path.join(output_dir, filename)
plt.tight_layout()
plt.subplots_adjust(bottom=0.1) # Espai per a la llegenda
plt.savefig(output_path, dpi=300, bbox_inches='tight')
print(f" Gràfic guardat a: {output_path}")
plt.show()

# Anàlisi d'estacionalitat
print("\n CONCLUSIONS D'ESTACIONALITAT:")

# 1. Categories amb major estacionalitat
seasonality_strength = (heatmap_data.max(axis=1) - heatmap_data.
˓→min(axis=1)).sort_values(ascending=False)
top_seasonal = seasonality_strength.head(5)
print("\n CATEGORIES AMB MAJOR ESTACIONALITAT:")
for cat, strength in top_seasonal.items():
    peak_month = heatmap_data.loc[cat].idxmax()
    print(f"\t{cat}: Variació {strength:.1f} (Pic a
˓→{MESOS_CATALA[peak_month]})")

# 2. Categories més estables
stability = heatmap_data.std(axis=1).sort_values()
top_stable = stability.head(5)
print("\n CATEGORIES MÉS ESTABLES:")
for cat, std_dev in top_stable.items():
    print(f"\t{cat}: Desviació {std_dev:.2f}")

# 3. Mesos amb més demanda global
monthly_avg = heatmap_data.mean(axis=0)
peak_months = monthly_avg.nlargest(3).index.tolist()
print("\n MESOS AMB MÉS DEMANDA GLOBAL:")

```

```

    for mes in peak_months:
        print(f"\n {MESOS_CATALA[mes]}: {monthly_avg[mes]:.1f} (sobre la mitjana)")

# 4. Impacte d'esdeveniments clau
print("\n IMPACTE D'ESDEVENIMENTS ESPECÍFICS:")
for mes, event in events.items():
    if mes in heatmap_data.columns:
        avg_impact = heatmap_data[mes].mean()
        print(f"\n {event} ({MESOS_CATALA[mes]}): {avg_impact:.1f}")

def analisi_preu_demanda(df, output_dir=None):
    print("\n" + "="*50)
    print(" ANÀLISI DE PREU I DEMANDA")
    start_time = time.time()

    if output_dir is None:
        output_dir = os.path.join(BASE_PATH, "visualitzacions")
    os.makedirs(output_dir, exist_ok=True)

    print(" Preparant dades per a l'anàlisi de preu i demanda...")
    cols_necessaries = [
        'product_id', 'categoria_catala', 'price',
        'order_item_id', 'order_id', 'review_score'
    ]
    df = df[cols_necessaries].copy()
    preu_original = len(df)
    df = df[df['price'] > 0]
    preu_filtrat = len(df)
    print(f" S'han filtrat {preu_original - preu_filtrat} registres amb preu menor o negatiu")
    df['price'] = df['price'].astype('float32')
    df['order_item_id'] = df['order_item_id'].astype('int16')
    df['review_score'] = df['review_score'].fillna(0).astype('float32')

    print(" Agregant dades per producte i categoria...")
    agg_step1 = df.groupby(['product_id'], as_index=False).agg(
        preu_mitja=('price', 'mean'),
        quantitat_total=('order_item_id', 'sum'),
        comandes_totals=('order_id', 'nunique'),
        mitjana_review=('review_score', 'mean')
    )
    categories = df[['product_id', 'categoria_catala']].drop_duplicates()
    df_agregat = pd.merge(agg_step1, categories, on='product_id', how='left')

    print(" Calculant rangs de preu i percentils...")
    bins = [0, 25, 50, 100, 200, 500, 1000, float('inf')]
```

```

    labels = ['<25 BRL', '25-50 BRL', '50-100 BRL', '100-200 BRL', '200-500 BRL', '500-1000 BRL', '>1000 BRL']
    df_agregat['rang_preu'] = pd.cut(df_agregat['preu_mitja'], bins=bins, labels=labels)
    df_agregat['percentil_preu'] = df_agregat.groupby('categoria_catala')['preu_mitja'].transform(lambda x: x.rank(pct=True))
    df_agregat['price_normalized'] = df_agregat.groupby('categoria_catala')['preu_mitja'].transform(lambda x: (x - x.min()) / (x.max() - x.min() + 1e-9))

    print(" Generant visualitzacions...")
    plt.figure(figsize=(18, 12))
    plt.suptitle('Anàlisi de Preu i Demanda', fontsize=20)

    plt.subplot(2, 2, 1)
    rang_demand = df_agregat.groupby('rang_preu')['quantitat_total'].mean().reset_index()
    ax = sns.barplot(
        x='rang_preu',
        y='quantitat_total',
        data=rang_demand,
        palette='viridis'
    )
    plt.title('Demanda Mitjana per Rang de Preu', fontsize=16)
    plt.xlabel('Rang de Preu', fontsize=14)
    plt.ylabel('Quantitat Venuda Mitjana', fontsize=14)
    plt.xticks(rotation=45, ha='right')
    for p in ax.patches:
        ax.annotate(f"{p.get_height():.1f}",
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='center', fontsize=10, color='black',
                    xytext=(0, 5),
                    textcoords='offset points')

    plt.subplot(2, 2, 2)
    sns.regplot(
        x='price_normalized',
        y='mitjana_review',
        data=df_agregat,
        scatter_kws={'s': 20, 'alpha': 0.3, 'color': '#3498db'},
        line_kws={'color': '#e74c3c', 'linewidth': 2.5},
        lowess=True
    )
    plt.title('Satisfacció vs. Posicionament de Preu', fontsize=16)
    plt.xlabel('Preu normalitzat dins la Categoria', fontsize=14)

```

```

plt.ylabel('Puntuació Mitjana de Resenyes', fontsize=14)
plt.ylim(0, 5.5)
plt.grid(True, alpha=0.2)

plt.subplot(2, 2, 3)
sns.kdeplot(
    data=df_agregat,
    x='preu_mitja',
    fill=True,
    common_norm=False,
    alpha=0.7,
    color='#2ecc71',
    log_scale=True
)
plt.title('Distribució de Preus (Escala Logarítmica)', fontsize=16)
plt.xlabel('Preu Mitjà (BRL)', fontsize=14)
plt.ylabel('Densitat', fontsize=14)

plt.subplot(2, 2, 4)
top_categories = df_agregat.groupby('categoria_catala')['price_normalized'].mean().nlargest(10)
top_categories.plot(kind='barh', color="#9b59b6")
plt.title('Top 10 Categories per Preu Relatiu Mig', fontsize=16)
plt.xlabel('Preu Normalitzat Mig (0=baix, 1=alt)', fontsize=14)
plt.ylabel('Categoria', fontsize=14)
plt.gca().invert_yaxis()

plt.tight_layout(rect=[0, 0, 1, 0.96])
output_path = os.path.join(output_dir, "analisi_preu_demanda.png")
plt.savefig(output_path, dpi=300, bbox_inches='tight')
print(f" Visualitzacions guardades a: {output_path}")
plt.show()

# =====
# INSIGHTS I RECOMANACIONS ESTRATÈGIQUES
# =====
print("\n" + "="*50)
print(" INSIGHTS CLAU I RECOMANACIONS ESTRATÈGIQUES")

# 1. Productes més venuts
top_products = df_agregat.nlargest(5, 'quantitat_total')[['categoria_catala', 'preu_mitja', 'quantitat_total']]
print("\n PRODUCTES MÉS VENUTS (per unitats):")
print(top_products.to_string(index=False))

# 2. Productes premiums amb alta demanda
high_end = df_agregat[

```

```

        (df_agregat['preu_mitja'] > 500) &
        (df_agregat['quantitat_total'] > df_agregat['quantitat_total'].median())
    ]
    if not high_end.empty:
        print("\n PRODUCTES PREMIUMS AMB ALTA DEMANDA:")
        print(high_end[['categoria_catala', 'preu_mitja', 'quantitat_total']]
              .sort_values('preu_mitja', ascending=False)
              .head(5)
              .to_string(index=False))

# 3. Productes amb alta valoració i baix preu
high_value = df_agregat[
    (df_agregat['preu_mitja'] < 50) &
    (df_agregat['mitjana_review'] > 4.5) &
    (df_agregat['quantitat_total'] > 10)
]
if not high_value.empty:
    print("\n PRODUCTES D'ALTA QUALITAT I BAIX PREU:")
    print(high_value[['categoria_catala', 'preu_mitja', 'mitjana_review', 'quantitat_total']]
          .sort_values('mitjana_review', ascending=False)
          .head(5)
          .to_string(index=False))

# =====
# RECOMANACIONS ESTRATÈGIQUES
# =====
print("\n RECOMANACIONS ESTRATÈGIQUES:")

# Recomanacions basades en productes més venuts
if not top_products.empty:
    top_cat = top_products['categoria_catala'].iloc[0]
    print("\n1. ESTRATÈGIA PER A PRODUCTES MASSIUS:")
    print(f" - Centrar-se en la categoria '{top_cat}' que representa el volum més alt de vendes")
    print(" - Optimitzar l'inventari i la logística per a aquests productes d'alta rotació")
    print(" - Crear paquets o ofertes combinades per augmentar la mitjana de venda")
    print(" - Utilitzar com a 'productes ganxo' per atraure nous clients")

# Recomanacions basades en productes premium
if not high_end.empty:
    print("\n2. ESTRATÈGIA PER A PRODUCTES PREMIUM:")
    print(" - Desenvolupar campanyes de màrqueting exclusives per aquests productes d'alta gamma")

```

```

        print("    - Millorar la presentació i descripcions d'aquests productes"
        ↪amb contingut multimèdia")
        print("    - Ofereir garanties esteses o serveis postvenda premium")
        print("    - Segmentar publicitat cap a clients d'alt poder adquisitiu")

# Recomanacions basades en productes d'alta valoració
if not high_value.empty:
    best_value = high_value.iloc[0]
    print("\n3. ESTRATÈGIA PER A PRODUCTES D'ALTA QUALITAT-PREU:")
    print(f"    - Promocionar agressivament productes com"
    ↪'{best_value['categoria_catala']} (BRL{best_value['preu_mitja']:.2f} -"
    ↪'{best_value['mitjana_review']}/5)")
    print("    - Destacar aquests productes a les pàgines d'inici i cerques")
    print("    - Utilitzar com a testimonis de qualitat de la botiga")
    print("    - Considerar augmentar lleugerament el preu sense superar el"
    ↪10%")

# Recomanacions generals
print("\n4. ESTRATÈGIES TRANSVERSALS:")
print("    - Implementar preus dinàmics basats en la demanda estacional")
print("    - Crear programes de fidelització per a compradors freqüents de"
    ↪products premium)
print("    - Fer tests A/B per a diferents estratègies de preu per"
    ↪categoria")
print("    - Desenvolupar bundles que combiniïn productes massius amb d'alt"
    ↪marge")

# Recomanacions específiques basades en dades
if not df_agregat.empty:
    high_margin_cats = df_agregat.groupby('categoria_catala')['preu_mitja'].mean().nlargest(3).index.tolist()
    print(f"\n5. OPORTUNITATS PER CATEGORIA:")
    print(f"    - Categories amb major marge: {', '.join(high_margin_cats)}")
    print(f"    - Considerar ampliar l'assortiment en aquestes categories")

end_time = time.time()
print(f"\n Temps d'execució de l'anàlisi: {end_time - start_time:.2f}"
    ↪segons")

return df_agregat

def analitzar_impacte_fotos_producte(df, output_dir=None):
    print("\n" + "="*50)
    print(" ANÀLISI DE L'IMPACTE DE LES FOTOS DEL PRODUCTE")
    start_time = time.time()

```

```

if output_dir is None:
    output_dir = os.path.join(BASE_PATH, "visualitzacions")
os.makedirs(output_dir, exist_ok=True)

df = df.copy()

df_agg = df.groupby('product_photos_qty').agg(
    num_productes=('product_id', 'nunique'),
    facturacio_total=('price', 'sum'),
    mitjana_review=('review_score', 'mean'),
    unitats_venues=('order_item_id', 'count')
).reset_index()

print("\n MÈTRIQUES CLAU PER QUANTITAT DE FOTOS:")
print(df_agg.to_string(index=False))

plt.figure(figsize=(18, 12))
plt.suptitle("Impacte de la Quantitat de Fotos en Mètriques Comercials", fontweight='bold', fontsize=20)

max_photos = int(df['product_photos_qty'].max())

ax1 = plt.subplot(2, 2, 1)
sns.regplot(
    x='product_photos_qty',
    y='facturacio_total',
    data=df_agg,
    scatter_kws={'s': 100, 'alpha': 0.6, 'color': '#1f77b4'},
    line_kws={'color': 'red', 'linewidth': 2},
    ax=ax1
)
ax1.set_title('Facturació Total per Quantitat de Fotos', fontsize=16)
ax1.set_xlabel('Quantitat de Fotos', fontsize=14)
ax1.set_ylabel('Facturació Total (log BRL)', fontsize=14)
ax1.set_yscale('log')
ax1.grid(True, alpha=0.2)
ax1.set_xticks(np.arange(0, max_photos+1, step=1))
ax1.set_xlim(-0.5, max_photos+0.5)

ax2 = plt.subplot(2, 2, 2)
sns.barplot(
    x='product_photos_qty',
    y='mitjana_review',
    data=df_agg,
    color='#9467bd',
    ax=ax2
)

```

```

ax2.set_title('Satisfacció Mitjana per Quantitat de Fotos', fontsize=16)
ax2.set_xlabel('Quantitat de Fotos', fontsize=14)
ax2.set_ylabel('Puntuació Mitjana de Resenyes', fontsize=14)
ax2.set_ylim(0, 5)
ax2.set_xticks(np.arange(0, max_photos+1))
ax2.set_xticklabels([str(int(x)) for x in np.arange(0, max_photos+1)])
mitjana_global = df['review_score'].mean()
ax2.axhline(y=mitjana_global, color='r', linestyle='--', alpha=0.7)
ax2.text(max_photos+0.5, mitjana_global+0.1, f'Mitjana global:{mitjana_global:.2f}', ha='right', color='r', fontsize=12)

ax3 = plt.subplot(2, 2, 3)
sns.regplot(
    x='product_photos_qty',
    y='unitats_venudes',
    data=df_agg,
    scatter_kws={'s': 100, 'alpha': 0.6, 'color': '#2ca02c'},
    line_kws={'color': 'red', 'linewidth': 2},
    ax=ax3
)
ax3.set_title('Unitats Venudes per Quantitat de Fotos', fontsize=16)
ax3.set_xlabel('Quantitat de Fotos', fontsize=14)
ax3.set_ylabel('Unitats Venudes', fontsize=14)
ax3.grid(True, alpha=0.2)
ax3.set_ylim(0, df_agg['unitats_venudes'].max() * 1.1)
ax3.set_xticks(np.arange(0, max_photos+1, step=1))
ax3.set_xlim(-0.5, max_photos+0.5)

ax4 = plt.subplot(2, 2, 4)
value_counts = df['product_photos_qty'].value_counts().sort_index()
for val, freq in value_counts.items():
    ax4.plot([val, val], [0, freq], color='#6A0DAD', linewidth=1.5, alpha=0.7)
    ax4.scatter(val, freq, s=80, color='#6A0DAD', edgecolor='black', zorder=5)
ax4.set_title('Distribució de la Quantitat de Fotos per Producte', fontsize=16)
ax4.set_xlabel('Quantitat de Fotos', fontsize=14)
ax4.set_ylabel('Freqüència', fontsize=14)
min_val = value_counts.index.min()
max_val = value_counts.index.max()
ax4.set_xlim(min_val - 0.5, max_val + 0.5)
ax4.set_ylim(0, value_counts.max() * 1.1)
step = 1 if (max_val - min_val) <= 20 else 2
ax4.set_xticks(np.arange(min_val, max_val+1, step))
ax4.set_xticklabels([str(int(x)) for x in ax4.get_xticks()])

```

```

mean_val = df['product_photos_qty'].mean()
median_val = df['product_photos_qty'].median()
ax4.axvline(mean_val, color='r', linestyle='--', alpha=0.7, label=f'Mitjana: {mean_val:.1f}')
ax4.axvline(median_val, color='g', linestyle='-', alpha=0.7, label=f'Mediana: {median_val}')
ax4.legend()

plt.tight_layout(rect=[0, 0, 1, 0.96])
output_path = os.path.join(output_dir, "impacte_fotos_producte.png")
plt.savefig(output_path, dpi=300, bbox_inches='tight')
print(f" Gràfic guardat a: {output_path}")
plt.show()

print("\n ANÀLISI DE CORRELACIÓ:")
correlacions = df[['product_photos_qty', 'price', 'review_score', 'order_item_id']].corr()
print(correlacions)

print("\n RECOMANACIONS ESTRATÈGIQUES:")
print("1. Productes amb 3-6 fotos tendeixen a tenir millors resultats comercials")
print("2. Un excés de fotos (>10) no millora la satisfacció del client")
print("3. Optimitzi la qualitat de les fotos en comptes de la quantitat")
print("4. Realitzi tests A/B per determinar el nombre òptim de fotos per categoria")

end_time = time.time()
print(f" Temps d'execució de l'anàlisi: {end_time - start_time:.2f} segons")

return df_agg

def analisi_preu_relatiu(df, output_dir=None):
    if output_dir is None:
        output_dir = os.path.join(BASE_PATH, "visualitzacions")

    print("\n" + "="*50)
    print(" ANÀLISI DETALLAT DE PREU RELATIU I SATISFACCIÓ")

    if 'price_normalized' not in df.columns or 'review_score' not in df.columns:
        print(" Dades insuficients per a l'anàlisi de preu relatiu")
        return

    plt.figure(figsize=(14, 7))
    sns.regplot(

```

```

        x='price_normalized',
        y='review_score',
        data=df,
        lowess=True,
        scatter_kws={'alpha': 0.2, 'color': '#3498db'},
        line_kws={'color': '#e74c3c', 'linewidth': 3}
    )
    plt.axhline(y=4.0, color='green', linestyle='--', alpha=0.5, label='Llindar de la Satisfacció')
    plt.title('Relació entre Preu Relatiu i Satisfacció del Client', fontsize=18)
    plt.xlabel('Preu Normalitzat (0 = més baix de la categoria, 1 = més alt)', fontsize=14)
    plt.ylabel('Puntuació de Resenya', fontsize=14)
    plt.legend()
    plt.grid(True, alpha=0.2)
    output_path = os.path.join(output_dir, "relacio_preu_satisfaccio.png")
    plt.savefig(output_path, dpi=300, bbox_inches='tight')
    print(f" Gràfic guardat a: {output_path}")
    plt.show()

df['quartil_preu'] = pd.qcut(df['price_normalized'], 4, labels=['Q1 (Més Baix)', 'Q2', 'Q3', 'Q4 (Més Alt)'])
satisfaccio_quartils = df.groupby('quartil_preu')['review_score'].mean().reset_index()
print("\n Satisfacció mitjana per quartil de preu relatiu:")
print(satisfaccio_quartils.to_string(index=False))

def mean_absolute_percentage_error_robust(y_true, y_pred, epsilon=1e-9):
    y_true = np.asarray(y_true)
    y_pred = np.asarray(y_pred)

    valid_mask = (y_true > epsilon) & (y_pred > 0)

    if np.sum(valid_mask) == 0:
        return float('inf')

    ape = np.abs((y_true[valid_mask] - y_pred[valid_mask]) / y_true[valid_mask])
    return np.mean(ape) * 100

def avaluar_model(y_real, y_pred):
    mae = mean_absolute_error(y_real, y_pred)
    mse = mean_squared_error(y_real, y_pred)
    rmse = np.sqrt(mse)
    mape = mean_absolute_percentage_error_robust(y_real, y_pred)
    r2 = r2_score(y_real, y_pred)
    medae = median_absolute_error(y_real, y_pred)

```

```

max_err = max_error(y_real, y_pred)

return {
    'MAE': mae,
    'MSE': mse,
    'RMSE': rmse,
    'MAPE': mape,
    'R22', float('nan')))
        rmse_vals.append(metrics.get('RMSE', float('nan'))))
        medae_vals.append(metrics.get('MedAE', float('nan'))))

    # Configurar el gràfic
    plt.figure(figsize=(18, 22))
    plt.suptitle(f'Mètriques de Rendiment per a les {top_n} Categories Principals', fontsize=24, y=0.98)

    # MAE
    plt.subplot(5, 1, 1)
    bars = plt.bar(categories, mae_vals, color="#3498db")
    plt.title('Error Absolut Mig (MAE) per Categoria', fontsize=18)
    plt.ylabel('MAE', fontsize=14)
    plt.grid(axis='y', alpha=0.3)
    plt.xticks(rotation=45, ha='right')

```

```

for bar in bars:
    yval = bar.get_height()
    if not np.isnan(yval):
        plt.text(bar.get_x() + bar.get_width()/2, yval + 0.5, f'{yval:.1f}', ha='center', va='bottom', fontsize=10)

# MAPE
plt.subplot(5, 1, 2)
bars = plt.bar(categories, mape_vals, color='#e74c3c')
plt.title('Error Percentual Absolut Mig (MAPE) per Categoria', fontsize=18)
plt.ylabel('MAPE (%)', fontsize=14)
plt.grid(axis='y', alpha=0.3)
plt.xticks(rotation=45, ha='right')
for bar in bars:
    yval = bar.get_height()
    if not np.isnan(yval):
        plt.text(bar.get_x() + bar.get_width()/2, yval + 0.5, f'{yval:..1f}%', ha='center', va='bottom', fontsize=10)

# R²
plt.subplot(5, 1, 3)
bars = plt.bar(categories, r2_vals, color="#2ecc71")
plt.title('Coeficient de Determinació (R²) per Categoria', fontsize=18)
plt.ylabel('R²', fontsize=14)
plt.grid(axis='y', alpha=0.3)
plt.xticks(rotation=45, ha='right')
for bar in bars:
    yval = bar.get_height()
    if not np.isnan(yval):
        plt.text(bar.get_x() + bar.get_width()/2, yval + 0.01, f'{yval:..2f}', ha='center', va='bottom', fontsize=10)

# RMSE
plt.subplot(5, 1, 4)
bars = plt.bar(categories, rmse_vals, color="#f39c12")
plt.title('Arrel de l\'Error Quadràtic Mig (RMSE) per Categoria', fontsize=18)
plt.ylabel('RMSE', fontsize=14)
plt.grid(axis='y', alpha=0.3)
plt.xticks(rotation=45, ha='right')
for bar in bars:
    yval = bar.get_height()
    if not np.isnan(yval):
        plt.text(bar.get_x() + bar.get_width()/2, yval + 0.5, f'{yval:.1f}', ha='center', va='bottom', fontsize=10)

```

```

# MedAE
plt.subplot(5, 1, 5)
bars = plt.bar(categories, medae_vals, color="#9b59b6")
plt.title('Error Absolut Median (MedAE) per Categoria', fontsize=18)
plt.ylabel('MedAE', fontsize=14)
plt.grid(axis='y', alpha=0.3)
plt.xticks(rotation=45, ha='right')
for bar in bars:
    yval = bar.get_height()
    if not np.isnan(yval):
        plt.text(bar.get_x() + bar.get_width()/2, yval + 0.5, f'{yval:.1f}', ha='center', va='bottom', fontsize=10)

plt.tight_layout(rect=[0, 0, 1, 0.96])

output_dir = os.path.join(BASE_PATH, "prediccions")
os.makedirs(output_dir, exist_ok=True)
output_path = os.path.join(output_dir, f"metrics_top_{top_n}_categories.{png}")
plt.savefig(output_path, dpi=300, bbox_inches='tight')
plt.show()

print(f" Gràfic de mètriques guardat a: {output_path}")

def visualitzar_correlacio_metrics(resultats):
    """Visualitza la correlació entre les mètriques de rendiment amb gestió de NaN"""
    if not resultats:
        print(" No hi ha resultats per visualitzar")
        return

    dades = []
    for cat, res in resultats.items():
        metrics = res.get('prophet_metrics', {})
        dades.append({
            'Categoria': cat,
            'MAE': metrics.get('MAE', float('nan')),
            'MAPE': metrics.get('MAPE', float('nan')),
            'R22', float('nan')),
            'RMSE': metrics.get('RMSE', float('nan')),
            'MedAE': metrics.get('MedAE', float('nan')),
            'MaxError': metrics.get('MaxError', float('nan'))
        })

    df_metrics = pd.DataFrame(dades).set_index('Categoria')
    df_metrics = df_metrics.dropna(how='all')

```

```

if df_metrics.empty:
    print(" No hi ha dades suficients per a la matriu de correlació")
    return

# Omplir valors NaN amb la mitjana
df_metrics_filled = df_metrics.fillna(df_metrics.mean())

corr = df_metrics_filled.corr()

plt.figure(figsize=(12, 10))
plt.title('Correlació entre Mètriques de Rendiment', fontsize=20)
sns.heatmap(
    corr,
    annot=True,
    fmt=".2f",
    cmap='coolwarm',
    center=0,
    linewidths=0.5,
    annot_kws={"size": 12}
)
plt.xticks(fontsize=12, rotation=45)
plt.yticks(fontsize=12, rotation=0)

output_dir = os.path.join(BASE_PATH, "prediccions")
os.makedirs(output_dir, exist_ok=True)
output_path = os.path.join(output_dir, "correlacio_metrics.png")
plt.savefig(output_path, dpi=300, bbox_inches='tight')
plt.show()
print(f" Matriu de correlació guardada a: {output_path}")

def comparar_top_pitjor_models(resultats, top5, pitjor5):
    """Compara els 3 models per a les categories destacades amb visualització clara"""
    models = ['prophet', 'xgboost', 'sarima']
    metrics = ['MAE', 'MAPE', 'R²']
    colors = ['#1f77b4', '#ff7f0e', '#2ca02c'] # Colors diferents per a cada model

    # Configuració de la visualització
    plt.figure(figsize=(20, 15))
    plt.suptitle('Comparativa Detallada de Models per Categories Destacades', fontsize=20, y=0.98)

    # Per a cada mètrica (MAE, MAPE, R²)
    for i, metric in enumerate(metrics, 1):
        # TOP 5 - Millors categories

```

```

plt.subplot(3, 2, 2*i-1)
for j, model in enumerate(models):
    # Recollir dades per a aquest model i mètrica
    values = []
    for categoria in top5['Categoria']:
        try:
            value = resultats[categoria][f'{model}_metrics'].get(metric, float('nan'))
            if np.isinf(value) or np.isnan(value):
                value = 0
            values.append(value)
        except KeyError:
            values.append(0)

    # Crear barres per a cada categoria
    x = np.arange(len(top5))
    plt.bar(x + j*0.25, values, width=0.25, color=colors[j], □
    label=f'{model.capitalize()}')

# Configurar eixos i llegendes
plt.title(f'TOP 5 Categories (Millor Prophet) - {metric}', fontsize=14)
plt.ylabel(metric)
plt.xticks(x + 0.25, top5['Categoria'], rotation=45, ha='right')
plt.grid(axis='y', alpha=0.2)
plt.legend()

# PITJOR 5 - Categories problemàtiques
plt.subplot(3, 2, 2*i)
for j, model in enumerate(models):
    values = []
    for categoria in pitjor5['Categoria']:
        try:
            value = resultats[categoria][f'{model}_metrics'].get(metric, float('nan'))
            if np.isinf(value) or np.isnan(value):
                value = 0
            values.append(value)
        except KeyError:
            values.append(0)

    x = np.arange(len(pitjor5))
    plt.bar(x + j*0.25, values, width=0.25, color=colors[j], □
    label=f'{model.capitalize()}')

    plt.title(f'PITJOR 5 Categories (Pitjor Prophet) - {metric}', □
    fontsize=14)
    plt.ylabel(metric)

```

```

plt.xticks(x + 0.25, pitjor5['Categoria'], rotation=45, ha='right')
plt.grid(axis='y', alpha=0.2)
plt.legend()

# Ajustar i guardar
plt.tight_layout(rect=[0, 0, 1, 0.96])
output_path = os.path.join(BASE_PATH, "prediccions",
↳"comparativa_top_pitjor_detall.png")
plt.savefig(output_path, dpi=300, bbox_inches='tight')
print(f" Comparativa detallada guardada a: {output_path}")
plt.show()

# Generar taula comparativa per a alimentar altres anàlisis
taula_comparativa = []

for categoria in top5['Categoria'].tolist() + pitjor5['Categoria'].tolist():
    fila = {'Categoria': categoria}
    for model in models:
        for metric in metrics:
            try:
                valor = resultats[categoria][f'{model}_metrics'].
↳get(metric, float('nan'))
                fila[f'{model}_{metric}'] = valor
            except KeyError:
                fila[f'{model}_{metric}'] = float('nan')
    taula_comparativa.append(fila)

df_comparatiu = pd.DataFrame(taula_comparativa)
return df_comparatiu

def comparar_models_en_categories(resultats_models, top_n=10):
    """Compara els models per a les categories principals amb gestió de dades
↳incompletes"""
    if not resultats_models:
        print(" No hi ha resultats per comparar")
        return None

    categories = list(resultats_models.keys())
    if len(categories) > top_n:
        categories = categories[:top_n]

    comparacio = []
    for cat in categories:
        res_cat = resultats_models[cat]

        # Prophet
        prophet_metrics = res_cat.get('prophet_metrics', {})

```

```

# XGBoost
xgboost_metrics = res_cat.get('xgboost_metrics', {})

# SARIMA
sarima_metrics = res_cat.get('sarima_metrics', {})

comparacio.append({
    'categoria': cat,
    'MAE_Prophet': prophet_metrics.get('MAE', float('nan')),
    'MAPE_Prophet': prophet_metrics.get('MAPE', float('nan')),
    'R2_Prophet': prophet_metrics.get('R2', float('nan')),
    'MAE_XGBoost': xgboost_metrics.get('MAE', float('nan')),
    'MAPE_XGBoost': xgboost_metrics.get('MAPE', float('nan')),
    'R2_XGBoost': xgboost_metrics.get('R2', float('nan')),
    'MAE_SARIMA': sarima_metrics.get('MAE', float('nan')),
    'MAPE_SARIMA': sarima_metrics.get('MAPE', float('nan')),
    'R2_SARIMA': sarima_metrics.get('R2', float('nan'))
})

df_comparacio = pd.DataFrame(comparacio)

# Configurar el gràfic
plt.figure(figsize=(16, 10))

# 1. Comparativa de MAE
plt.subplot(3, 1, 1)
df_mae = df_comparacio[['categoria', 'MAE_Prophet', 'MAE_XGBoost', ↴
    'MAE_SARIMA']].set_index('categoria')
df_mae.plot(kind='bar', ax=plt.gca())
plt.title('Comparativa de MAE per Model i Categoria', fontsize=16)
plt.ylabel('MAE', fontsize=12)
plt.legend(title='Model', loc='upper right')
plt.grid(axis='y', alpha=0.3)

# 2. Comparativa de MAPE
plt.subplot(3, 1, 2)
df_mape = df_comparacio[['categoria', 'MAPE_Prophet', 'MAPE_XGBoost', ↴
    'MAPE_SARIMA']].set_index('categoria')
df_mape.plot(kind='bar', ax=plt.gca())
plt.title('Comparativa de MAPE per Model i Categoria', fontsize=16)
plt.ylabel('MAPE (%)', fontsize=12)
plt.legend(title='Model', loc='upper right')
plt.grid(axis='y', alpha=0.3)

# 3. Comparativa de R2
plt.subplot(3, 1, 3)

```

```

df_r2 = df_comparacio[['categoria', 'R2_Prophet', 'R2_XGBoost', 'R2_SARIMA']].set_index('categoria')
df_r2.plot(kind='bar', ax=plt.gca())
plt.title('Comparativa de R2 per Model i Categoria', fontsize=16)
plt.ylabel('R2', fontsize=12)
plt.legend(title='Model', loc='lower right')
plt.grid(axis='y', alpha=0.3)

plt.tight_layout()

# Guardar el resultat
output_dir = os.path.join(BASE_PATH, "prediccions")
os.makedirs(output_dir, exist_ok=True)
output_path = os.path.join(output_dir, "comparacio_models_categories.png")
plt.savefig(output_path, dpi=300, bbox_inches='tight')
plt.show()
print(f" Comparativa de models guardada a: {output_path}")

return df_comparacio

def generar_resum_comparatiu(resultats):
    """Genera un resum comparatiu del rendiment de tots els models"""
    if not resultats:
        print(" No hi ha resultats per mostrar")
        return

    # Preparar dades per al gràfic comparatiu
    categories = list(resultats.keys())

    # Extreure mètriques de l'estructura niada
    mae_values = []
    mape_values = []
    r2_values = []

    for cat in categories:
        metrics = resultats[cat].get('prophet_metrics', {})
        mae_values.append(metrics.get('MAE', float('nan')))
        mape_values.append(metrics.get('MAPE', float('nan'))))
        r2_values.append(metrics.get('R2', float('nan'))))

    # Crear gràfic amb 3 subplots verticals
    plt.figure(figsize=(14, 16))

    # 1. MAE
    plt.subplot(3, 1, 1)
    bars_mae = plt.bar(categories, mae_values, color='skyblue')
    plt.title('Comparació de MAE per Categoria', fontsize=16)

```

```

plt.ylabel('MAE (Error Absolut Mig)', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.3)

# Afegir valors a les barres MAE
for bar in bars_mae:
    height = bar.get_height()
    if not np.isnan(height):
        plt.text(bar.get_x() + bar.get_width()/2., height,
                  f'{height:.1f}',
                  ha='center', va='bottom',
                  fontsize=10)

# 2. MAPE
plt.subplot(3, 1, 2)
bars_mape = plt.bar(categories, mape_values, color='salmon')
plt.title('Comparació de MAPE per Categoria', fontsize=16)
plt.ylabel('MAPE (%)', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.3)

# Afegir valors a les barres MAPE
for bar in bars_mape:
    height = bar.get_height()
    if not np.isnan(height):
        plt.text(bar.get_x() + bar.get_width()/2., height,
                  f'{height:.1f}%',
                  ha='center', va='bottom',
                  fontsize=10)

# 3. R2
plt.subplot(3, 1, 3)
bars_r2 = plt.bar(categories, r2_values, color="#2ecc71")
plt.title('Comparació de R2 per Categoria', fontsize=16)
plt.ylabel('Coeficient de Determinació (R2)', fontsize=12)
plt.ylim(0, 1)
# plt.axhline(y=0.7, color='r', linestyle='--', alpha=0.7)

# Afegir valors a les barres R2
for bar in bars_r2:
    height = bar.get_height()
    if not np.isnan(height):
        plt.text(bar.get_x() + bar.get_width()/2., height,
                  f'{height:.2f}',
                  ha='center', va='bottom',
                  fontsize=10)

# Configuració comuna per tots els subplots
for i in range(1, 4):

```

```

plt.subplot(3, 1, i)
plt.xticks(rotation=45, ha='right', fontsize=10)

plt.tight_layout(pad=3.0)

# Guardar resum
output_dir = os.path.join(BASE_PATH, "prediccions")
os.makedirs(output_dir, exist_ok=True)
output_path = os.path.join(output_dir, "resum_comparatiu_categories.png")
plt.savefig(output_path, dpi=300, bbox_inches='tight')
print(f" Gràfic guardat a: {output_path}")

# Mostrar el gràfic
plt.show()

print(f"\n RESUM COMPARATIU GUARDAT I MOSTRAT")
print("=". * 50)
print("RENDIMENT PER CATEGORIA:")
for cat in categories:
    metrics = resultats[cat].get('prophet_metrics', {})
    print(f"- {cat}: MAE = {metrics.get('MAE', 'N/A'):.2f}, MAPE = {metrics.
        get('MAPE', 'N/A'):.2f}%, R2 = {metrics.get('R2', 'N/A'):.4f}")

def visualitzar_top_pitjor_models(resultats_models, output_dir=None):
    if not resultats_models:
        print(" No hi ha resultats per mostrar")
        return None, None

    if output_dir is None:
        output_dir = os.path.join(BASE_PATH, "prediccions")
    os.makedirs(output_dir, exist_ok=True)

    categories = list(resultats_models.keys())
    mape_values = []
    for cat in categories:
        metrics = resultats_models[cat].get('prophet_metrics', {})
        mape = metrics.get('MAPE', float('nan'))
        mape_values.append(mape)

    df_models = pd.DataFrame({
        'Categoria': categories,
        'MAPE': mape_values
    }).sort_values('MAPE')

    top5 = df_models.head(5)
    pitjor5 = df_models.tail(5)

```

```

plt.figure(figsize=(14, 7))
ax = sns.barplot(
    x='Categoria',
    y='MAPE',
    data=top5,
    palette='viridis'
)
plt.title('Top 5 Models Predictius (Menor Error)', fontsize=18)
plt.xlabel('Categoria', fontsize=14)
plt.ylabel('MAPE (%)', fontsize=14)
plt.xticks(rotation=45)
plt.grid(axis='y', alpha=0.3)
for p in ax.patches:
    ax.annotate(f"{p.get_height():.2f}%",
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', fontsize=12, color='black', ↴
                xytext=(0, 10),
                textcoords='offset points')
output_path_top = os.path.join(output_dir, "top5_millors_models.png")
plt.savefig(output_path_top, dpi=300, bbox_inches='tight')
print(f" Gràfic dels 5 millors models guardat a: {output_path_top}")
plt.show()

plt.figure(figsize=(14, 7))
ax = sns.barplot(
    x='Categoria',
    y='MAPE',
    data=pitjor5,
    palette='rocket'
)
plt.title('Top 5 Models Predictius (Major Error)', fontsize=18)
plt.xlabel('Categoria', fontsize=14)
plt.ylabel('MAPE (%)', fontsize=14)
plt.xticks(rotation=45)
plt.grid(axis='y', alpha=0.3)
for p in ax.patches:
    ax.annotate(f"{p.get_height():.2f}%",
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', fontsize=12, color='black', ↴
                xytext=(0, 10),
                textcoords='offset points')
output_path_pitjor = os.path.join(output_dir, "top5_pitjors_models.png")
plt.savefig(output_path_pitjor, dpi=300, bbox_inches='tight')
print(f" Gràfic dels 5 pitjors models guardat a: {output_path_pitjor}")
plt.show()

return top5, pitjor5

```

```

def mostrar_prediccions_top_models(resultats_models, top5, output_dir=None):
    if output_dir is None:
        output_dir = os.path.join(BASE_PATH, "prediccions", "categories")

    if not os.path.exists(output_dir) or top5.empty:
        print(" No hi ha dades per mostrar")
        return

    plt.figure(figsize=(18, 12))
    plt.suptitle('Prediccions dels 5 Millors Models', fontsize=24, y=0.98)

    for i, (_, row) in enumerate(top5.iterrows(), 1):
        categoria = row['Categoria']
        img_path = os.path.join(output_dir, f"prediccions_{categoria}.png")

        if os.path.exists(img_path):
            ax = plt.subplot(5, 1, i)
            img = plt.imread(img_path)
            ax.imshow(img)
            ax.axis('off')
            ax.set_title(f"{categoria} (MAPE: {row['MAPE']:.2f}%)", ↴
             fontsize=14, pad=10)

    plt.tight_layout(rect=[0, 0, 1, 0.96])
    plt.show()

def mostrar_prediccions_pitjor_models(resultats_models, pitjor5, ↴
                                         output_dir=None):
    if output_dir is None:
        output_dir = os.path.join(BASE_PATH, "prediccions", "categories")

    if not os.path.exists(output_dir) or pitjor5.empty:
        print(" No hi ha dades per mostrar")
        return

    plt.figure(figsize=(18, 12))
    plt.suptitle('Prediccions dels 5 Pitjors Models', fontsize=24, y=0.98)

    for i, (_, row) in enumerate(pitjor5.iterrows(), 1):
        categoria = row['Categoria']
        img_path = os.path.join(output_dir, f"prediccions_{categoria}.png")

        if os.path.exists(img_path):
            ax = plt.subplot(5, 1, i)
            img = plt.imread(img_path)
            ax.imshow(img)

```

```

        ax.axis('off')
        ax.set_title(f"[{categoria}] (MAPE: {row['MAPE']:.2f}%)",□
        fontsize=14, pad=10)

    plt.tight_layout(rect=[0, 0, 1, 0.96])
    plt.show()

def get_periode_comercial(date):
    date = pd.Timestamp(date)
    if date >= pd.Timestamp('2016-09-04') and date < pd.Timestamp('2017-09-04'):
        return '2016-2017'
    elif date >= pd.Timestamp('2017-09-04') and date < pd.
    ↪Timestamp('2018-09-04'):
        return '2017-2018'
    elif date >= pd.Timestamp('2018-09-04') and date < pd.
    ↪Timestamp('2019-09-04'):
        return '2018-2019'
    elif date >= pd.Timestamp('2019-09-04'):
        return '2019-post'
    else:
        return 'fora_de_periode'

def verificar_serie(serie, categoria):
    if len(serie) < 52:
        print(f" [{categoria}]: Només {len(serie)} punts, mínim 52 requerits")
        return False
    if np.std(serie) < 1e-5:
        print(f" [{categoria}]: Variància gairebé zero")
        return False
    if np.any(serie < 0):
        print(f" [{categoria}]: Té valors negatius")
    return True

def preparar_dades_per_model(df):
    if 'categoria_catala' not in df.columns:
        print(" La columna 'categoria_catala' és necessària")
        return None

    print("\n Preparant dades per a model predictiu...")

    df['any_iso'] = df['order_purchase_timestamp'].dt.isocalendar().year
    df['setmana_iso'] = df['order_purchase_timestamp'].dt.isocalendar().week
    df['setmana'] = df['any_iso'].astype(str) + '-' + df['setmana_iso'].
    ↪astype(str)

    df_model = df.groupby(['setmana', 'categoria_catala']).agg(
        vendes=('order_id', 'nunique'),

```

```

    ingressos=('price', 'sum'),
    satisfaccio=('review_score', 'mean')
).reset_index()

df_model[['any', 'num_setmana']] = df_model['setmana'].str.split('-',  

→expand=True)
df_model['any'] = df_model['any'].astype(int)
df_model['num_setmana'] = df_model['num_setmana'].astype(int)

df_pivot = df_model.pivot_table(  

    index=['any', 'num_setmana'],
    columns='categoria_catala',
    values='vendes',
    fill_value=0
).reset_index()

print(f" Dades preparades amb {len(df_pivot)} setmanes i {len(df_pivot.  

→columns)-2} categories")
return df_pivot

def entrenar_model_prophet(df, categoria, periods=12):
    try:
        print(f" Entrenant model Prophet per a {categoria}...")
        start_time = time.time()

        # 1. Preparar dades per a Prophet
        df = df.copy()
        df['y'] = df[categoria]
        df['ds'] = df.apply(
            lambda row: datetime.fromisocalendar(int(row['any']),  

→int(row['num_setmana']), 1),
            axis=1
        )

        # 2. Afegir variables de període comercial
        df['periode_comercial'] = df['ds'].apply(get_periode_comercial)
        periodes_dummies = pd.get_dummies(df['periode_comercial'],  

→prefix='periode')
        df = pd.concat([df, periodes_dummies], axis=1)
        regressors = periodes_dummies.columns.tolist()

        # 3. Crear i configurar model (amb interval del 95%)
        model = Prophet(
            interval_width=0.95, # Interval de confiança del 95%
            yearly_seasonality=True,
            weekly_seasonality=True,
            daily_seasonality=False,

```

```

        seasonality_mode='multiplicative'
    )

# 4. Afegir regressors
for reg in regressors:
    model.add_regressor(reg)

# 5. Entrenar model
model.fit(df[['ds', 'y']] + regressors)

# 6. Crear futur
future = model.make_future_dataframe(periods=periods, freq='W')
future['periode_comercial'] = future['ds'].apply(get_periode_comercial)
future_dummies = pd.get_dummies(future['periode_comercial'],□
prefix='periode')
future = pd.concat([future, future_dummies], axis=1)

# 7. Assegurar tots els regressors
for reg in regressors:
    if reg not in future:
        future[reg] = 0

# 8. Fer prediccions
forecast = model.predict(future)

# 9. Avaluar model
merged = pd.merge(df, forecast[['ds', 'yhat']], on='ds', how='inner')
avaluacio = avaluar_model(merged['y'], merged['yhat'])

temp_exec = time.time() - start_time
print(f" Model entrenat per a {categoria} en {temp_exec:.2f}s")
print(f" - MAE: {avaluacio['MAE']:.2f}, MAPE: {avaluacio['MAPE']:.2f}%, R2: {avaluacio['R2']:.4f}")
print(f" - MSE: {avaluacio['MSE']:.2f}, RMSE: {avaluacio['RMSE']:.2f}, MedAE: {avaluacio['MedAE']:.2f}, MaxError: {avaluacio['MaxError']:.2f}")

# 10. Generar visualització
try:
    visualitzar_prediccions_per_categoria(model, forecast, df,□
categoria)
except Exception as e:
    print(f" Error en generar gràfic per a {categoria}: {str(e)}")
    traceback.print_exc()

return {
    'model': model,
    'forecast': forecast,
}

```

```

        'avaluacio': avaluacio,
        'tempo': temps_exec,
        'dades_entrenament': df
    }

except Exception as e:
    print(f" Error en entrenar model per a {categoria}: {str(e)}")
    traceback.print_exc()
    return None

def visualitzar_prediccions_per_categoria(model, forecast, df_real, categoria):
    # Filtrar dades fins a setembre 2018
    cutoff_date = pd.Timestamp('2018-09-30')
    mask_real = df_real['ds'] <= cutoff_date
    mask_forecast = forecast['ds'] <= cutoff_date

    plt.figure(figsize=(14, 7))

    # Dades reals (fins a setembre 2018)
    plt.plot(df_real.loc[mask_real, 'ds'],
              df_real.loc[mask_real, 'y'],
              'bo-', label='Dades Reals', alpha=0.7, markersize=4)

    # Prediccions (fins a setembre 2018)
    plt.plot(forecast.loc[mask_forecast, 'ds'],
              forecast.loc[mask_forecast, 'yhat'],
              'r--', label='Prediccions', linewidth=2)

    # Interval de confiança (fins a setembre 2018)
    plt.fill_between(forecast.loc[mask_forecast, 'ds'],
                     forecast.loc[mask_forecast, 'yhat_lower'],
                     forecast.loc[mask_forecast, 'yhat_upper'],
                     color='gray',
                     alpha=0.2,
                     label='Interval de Confiança (95%)')

    # Línia vertical per a setembre 2018
    plt.axvline(x=cutoff_date, color='gray', linestyle='--', alpha=0.7)
    plt.text(cutoff_date, plt.ylim()[1]*0.95, 'Set 2018',
             rotation=90, verticalalignment='top', fontsize=12)

    mean_lower = forecast.loc[mask_forecast, 'yhat_lower'].mean()
    mean_upper = forecast.loc[mask_forecast, 'yhat_upper'].mean()
    plt.text(0.02, 0.95,
             f"Interval mitjà: ({mean_lower:.1f} - {mean_upper:.1f})",
             transform=plt.gca().transAxes,

```

```

        fontsize=10,
        bbox=dict(facecolor='white', alpha=0.7))

plt.title(f'Prediccions vs Reals: {categoria}', fontsize=16)
plt.xlabel('Data')
plt.ylabel('Vendes')
plt.legend()
plt.grid(True, alpha=0.3)
plt.xticks(rotation=45)

output_dir = os.path.join(BASE_PATH, "prediccions", "categories")
os.makedirs(output_dir, exist_ok=True)
output_path = os.path.join(output_dir, f"prediccions_{categoria}.png")
plt.savefig(output_path, dpi=300, bbox_inches='tight')
plt.close()
print(f" Gràfic guardat a: {output_path}")

def entrenar_i_avaluar_prophet(df, categoria):
    result = entrenar_model_prophet(df, categoria)
    if result is None:
        return None, {}
    return result['model'], result['avaluacio']

def entrenar_i_avaluar_xgboost(df, categoria):
    """Entrena i avalua un model XGBoost amb validació temporal"""
    try:
        print(f" Entrenant XGBoost per a {categoria}...")

        # Preparar dades
        X = df[['any', 'num_setmana']]
        y = df[categoria]

        # Validació temporal (train: 80% inicial, test: 20% final)
        split_idx = int(len(X) * 0.8)
        X_train, X_test = X[:split_idx], X[split_idx:]
        y_train, y_test = y[:split_idx], y[split_idx:]

        # Entrenar model
        model = XGBRegressor(objective='reg:squarederror',
                              n_estimators=200,
                              learning_rate=0.1,
                              max_depth=3)
        model.fit(X_train, y_train)

        # Prediccions
        y_pred = model.predict(X_test)

```

```

# Avaluar només en el conjunt de prova
avaluacio = avaluar_model(y_test, y_pred)
print(f" XGBoost entrenat per a {categoria}")
print(f" - MAE: {avaluacio['MAE']:.2f}, MAPE: {avaluacio['MAPE']:.2f}%, R2: {avaluacio['R2']:.4f}")

return model, avaluacio

except Exception as e:
    print(f" Error en XGBoost per a {categoria}: {str(e)}")
    traceback.print_exc()
    return None, {}

def entrenar_i_avaluar_sarima(df, categoria):
    """Entrena i avalia un model SARIMA bàsic"""
    try:
        print(f" Entrenant SARIMA per a {categoria}...")

        # Preparar sèrie temporal
        serie = df[categoria]

        # Entrenar model (paràmetres bàsics)
        model = SARIMAX(
            serie,
            order=(1, 0, 0),
            seasonal_order=(0, 0, 0, 0),
            enforce_stationarity=False,
            enforce_invertibility=False
        )
        model_fit = model.fit(disp=False)

        # Prediccions
        y_pred = model_fit.predict()

        # Avaluar
        avaluacio = avaluar_model(serie, y_pred)
        print(f" SARIMA entrenat per a {categoria}")
        print(f" - MAE: {avaluacio['MAE']:.2f}, MAPE: {avaluacio['MAPE']:.2f}%, R2: {avaluacio['R2']:.4f}")

        return model_fit, avaluacio

    except Exception as e:
        print(f" Error en SARIMA per a {categoria}: {str(e)}")
        return None, {}

```

```

def entrenar_models_per_totes_categories(df, top_n=15):
    # Identificar columnes no de categoria
    non_category_cols = ['any', 'num_setmana']
    # Filtrar només les columnes que existeixen
    non_category_cols = [col for col in non_category_cols if col in df.columns]
    category_cols = [col for col in df.columns if col not in non_category_cols]

    if not category_cols:
        print(" No s'han trobat columnes de categories")
        return {}

    # Calcular vendes totals per categoria
    total_sales = df[category_cols].sum()
    top_categories = total_sales.nlargest(top_n).index.tolist()

    resultats = {}
    for category in tqdm(top_categories, desc="Entrenant models per categoria"):
        try:
            # Assegurar estructura bàsica
            resultats[category] = {
                'prophet_metrics': {},
                'xgboost_metrics': {},
                'sarima_metrics': {},
                'error': None
            }

            # Prophet
            try:
                model_prophet, metrics_prophet = entrenar_i_avalar_prophet(df, ↵
                ↵category)
                resultats[category]['prophet_metrics'] = metrics_prophet
                resultats[category]['model_prophet'] = model_prophet
            except Exception as e:
                print(f" Error Prophet per a {category}: {str(e)}")
                resultats[category]['error'] = str(e)

            # XGBoost
            try:
                model_xgboost, metrics_xgboost = entrenar_i_avalar_xgboost(df, ↵
                ↵category)
                resultats[category]['xgboost_metrics'] = metrics_xgboost
                resultats[category]['model_xgboost'] = model_xgboost
            except Exception as e:
                print(f" Error XGBoost per a {category}: {str(e)}")

            # SARIMA
            try:

```

```

        model_sarima, metrics_sarima = entrenar_i_avaluar_sarima(df, ↴
category)
        resultats[category]['sarima_metrics'] = metrics_sarima
        resultats[category]['model_sarima'] = model_sarima
    except Exception as e:
        print(f" Error SARIMA per a {category}: {str(e)}")

    # Verificar dades mínimes
    for model_type in ['prophet', 'xgboost', 'sarima']:
        metrics = resultats[category].get(f'{model_type}_metrics', {})
        if not metrics or 'MAE' not in metrics:
            print(f" {model_type.capitalize()} sense mètriques per a ↴
{category}")
        resultats[category][f'{model_type}_metrics'] = {
            'MAE': float('nan'),
            'MAPE': float('nan'),
            'R²': float('nan')
        }

    except Exception as e:
        print(f" Error crític per a {category}: {str(e)}")
        traceback.print_exc()

    return resultats

# =====
# 5. EXECUCIÓ PRINCIPAL I ANÀLISI
# =====

if __name__ == '__main__':
    print("=="*100)
    print(" EXECUCIÓ PRINCIPAL: PREPROCESSAMENT I ANÀLISI")
    print("=="*100)
    start_total = time.time()

    try:
        # -----
        print("\n" + "=="*100)
        print(" FASE 1: PREPROCESSAMENT DE DADES")
        # -----

        dfs = {}
        for nom, arxiu in ARXIUS.items():
            ruta = os.path.join(BASE_PATH, arxiu)
            print(f" Carregant {arxiu}...")
            try:
                dfs[nom] = pd.read_csv(ruta)
                dfs[nom] = convertir_timestamps(dfs[nom], nom)

```

```

        print(f" {arxiu} carregat i timestamps convertits")
    except Exception as e:
        print(f" ERROR carregant {arxiu}: {str(e)}")
        dfs = None
        break

    if dfs is None:
        print(" ERROR: No es pot continuar sense dades bàsiques")
        exit(1)

    print("\n" + "="*80)
    print(" VERIFICANT RELACIONS I QUALITAT DE DADES")
    verificar_relacions_i_qualitat(dfs['orders'], dfs['payments'], dfs)

    print("\n Processant categories de productes...")
    try:
        dfs['products'] = dfs['products'].merge(
            dfs['traduccio'],
            on='product_category_name',
            how='left'
        )
        n_abans = len(dfs['products'])
        dfs['products'] = dfs['products'].dropna(subset=['product_category_name_english'])
        n_despres = len(dfs['products'])
        print(f" S'han eliminat {n_abans - n_despres} productes sense traducció a l'anglès")
        dfs['products']['categoria_catala'] = (
            dfs['products']['product_category_name_english']
            .map(lambda x: TRADUCCIONS_CA.get(x, 'sense_traduccio'))
        )
        sense_traduccio_post = len(dfs['products'][dfs['products']['categoria_catala'] == 'sense_traduccio'])
        print(f" Productes sense traducció després del neteja: {len(sense_traduccio_post)}")
    except Exception as e:
        print(f" ERROR en processar categories: {str(e)}")
        dfs = None

    if dfs is None:
        print(" ERROR: Problemes greus en processar categories")
        exit(1)

    print("\n Fusionant taules per crear el dataset final...")
    try:
        df_final = (
            dfs['orders']

```

```

    .merge(dfs['customers'], on='customer_id', how='left')
    .merge(dfs['order_items'], on='order_id', how='left')
    .merge(dfs['products'][[
        'product_id', 'categoria_catala',
        'product_photos_qty', 'product_weight_g',
        'product_length_cm', 'product_height_cm', 'product_width_cm'
    ]], on='product_id', how='left')
    .merge(dfs['reviews'].groupby('order_id')['review_score'].
    ↪mean().reset_index(),
           on='order_id', how='left')
    .merge(dfs[['seller_id', 'seller_city', ↪
    ↪'seller_state']], on='seller_id', how='left')
)
del dfs
gc.collect()
print(" Memòria alliberada de datasets intermedis")
except Exception as e:
    print(f" ERROR en fusionar taules: {str(e)}")
df_final = None

if df_final is None:
    print(" ERROR: No es pot crear el dataset final")
    exit(1)

print("\n" + "="*50)
print(" INVESTIGANT PREUS INVÀLIDS (0 O NEGATIUS)")
try:
    invalid_prices = df_final[df_final['price'] <= 0]
    num_invalid = len(invalid_prices)
    if num_invalid > 0:
        print(f" ALERTA: S'han detectat {num_invalid} registres amb
    ↪preus no positius")
        print("\n MOSTRA DE REGISTRES INVÀLIDS (5 aleatoris):")
        print(invalid_prices.sample(min(5, num_invalid), ↪
    ↪random_state=42)[[
        'order_id', 'product_id', 'price', 'categoria_catala'
    ]].to_string(index=False))
        print("\n ANÀLISI DELS PREUS INVÀLIDS:")
        print(f"• Preu més baix: {invalid_prices['price'].min()} BRL")
        print(f"• Preu mitjà: {invalid_prices['price'].mean():.2f} BRL")
        print(f"• Ordres úniques afectades: {invalid_prices['order_id'].nunique()}")
        print(f"• Productes únics afectats: {invalid_prices['product_id'].nunique()}")

```

```

        print(f"• Categories afectades: {', '.join(invalid_prices['categoria_catala'].unique())}")
        if 'freight_value' in invalid_prices.columns:
            print(f"• Enviaments amb cost 0: {invalid_prices['freight_value'] == 0}.sum()")
            print(f"• Enviaments amb cost >0: {invalid_prices['freight_value'] > 0}.sum()")
        if 'order_status' in invalid_prices.columns:
            print("Estat de les comandes:")
            print(invalid_prices['order_status'].value_counts())
        print("\n ELIMINANT REGISTRES INVÀLIDS...")
        original_size = len(df_final)
        df_final = df_final[df_final['price'] > 0]
        new_size = len(df_final)
        print(f" S'han eliminat {original_size - new_size} registres invàlids")
    print(f" Nou tamany del dataset: {new_size:,} registres")
else:
    print(" No s'han detectat preus invàlids")
except Exception as e:
    print(f" AVÍS en verificar preus: {str(e)}")

print("\n" + "="*50)
print("GESTIÓ DE VALORS NULS EN TIMESTAMPES")
try:
    df_final = gestionar_valors_nuls_temporals(df_final)
    print(" Valors nuls en timestamps gestionats")
except Exception as e:
    print(f" AVÍS en gestionar valors nuls temporals: {str(e)}")

print("\n" + "="*50)
print("NORMALITZACIÓ DE PREUS PER CATEGORIA")
try:
    df_final = afegir_normalitzacio_preus(df_final)
    print(" Preus normalitzats per categoria")
except Exception as e:
    print(f" AVÍS en normalitzar preus: {str(e)}")

print("\n" + "="*80)
print(" OPTIMITZACIÓ DE MEMÒRIA")
try:
    df_final = optimitzar_memoria(df_final)
    print(" Memòria optimitzada")
except Exception as e:
    print(f" AVÍS en optimitzar memòria: {str(e)}")

print("\n" + "="*80)

```

```

print(" AFEGINT VARIABLES TEMPORALS")
try:
    df_final = afegir_variables_temporals(df_final)
    print(" Variables temporals afegides")
except Exception as e:
    print(f" AVÍS en afegir variables temporals: {str(e)}")

print("\n" + "="*50)
print(" DATAFRAME FINAL PREPARAT")
print(f"Registres: {len(df_final)}")
print(f"Columnes: {len(df_final.columns)}")
print(f"Ús de memòria: {df_final.memory_usage(deep=True).sum() / (1024**2):.2f} MB")

EXPORT_DIR = os.path.join(BASE_PATH, "preprocessat")
os.makedirs(EXPORT_DIR, exist_ok=True)
output_path = os.path.join(EXPORT_DIR, "ecommerce_brasil_preprocessat.parquet")
df_final.to_parquet(output_path, engine='pyarrow', index=False)
print(f"\n" + "="*50)
print(f" Dataset exportat a: {output_path}")
generar_informe_final(df_final, output_path)
end_preprocess = time.time()
print(f"\n Temps d'execució del preprocessament: {end_preprocess - start_total:.2f} segons")

# -----
print("\n" + "-"*100)
print(" FASE 2: ANÀLISI EXPLORATÒRIA")
analisis_start = time.time()

# -----
print("\n" + "-"*50)
print(" FASE 2.1. ANÀLISI DE FACTURACIÓ MENSUAL EN BRL BRASIL")
try:
    facturacio_mensual = plot_facturacio_mensual(df_final)
    print(f" Facturació mensual analitzada. Registres: {len(facturacio_mensual)} mesos")
except Exception as e:
    print(f" ERROR en anàlisi de facturació: {str(e)}")
# -----

# -----
print("\n" + "-"*50)
print(" FASE 2.2. TOP CATEGORIES PER FACTURACIÓ")
try:
    visualitzacions_dir = os.path.join(BASE_PATH, "visualitzacions")

```

```

os.makedirs(visualitzacions_dir, exist_ok=True)

print("\n FASE 2.2.1. TOP CATEGORIES PER FACTURACIÓ 2016-2017")
top_2016_2017 = plot_top10_periode_2016_2017(df_final, ↵
visualitzacions_dir)

print("\n FASE 2.2.2. TOP CATEGORIES PER FACTURACIÓ 2017-2018")
top_2017_2018 = plot_top10_periode_2017_2018(df_final, ↵
visualitzacions_dir)
except Exception as e:
    print(f" ERROR en anàlisi de categories: {str(e)}")
# ----

# -----
print("\n" + "-"*50)
print(" FASE 2.4. TOP CATEGORIES PER UNITATS")
try:
    print("\n FASE 2.4.1. TOP CATEGORIES PER UNITATS 2016-2017")
    top_units_2016_2017 = plot_top10_periode_2016_2017_units(df_final, ↵
visualitzacions_dir)

    print("\n FASE 2.4.2. TOP CATEGORIES PER UNITATS 2017-2018")
    top_units_2017_2018 = plot_top10_periode_2017_2018_units(df_final, ↵
visualitzacions_dir)
except Exception as e:
    print(f" ERROR en anàlisi de categories per unitats: {str(e)}")
# ----

# -----
print("\n" + "-"*50)
print(" FASE 2.3. COMPARATIVA PERÍODES COMERCIALS")
try:
    print("\n FASE 2.3.1. COMPARATIVA FACTURACIÓ PERÍODES COMERCIALS")
    plot_comparativa_facturacio_periodes(top_2016_2017, top_2017_2018, ↵
visualitzacions_dir)

    print("\n FASE 2.3.2. COMPARATIVA UNITATS PERÍODES COMERCIALS")
    plot_comparativa_unitats_periodes(top_units_2016_2017, ↵
top_units_2017_2018, visualitzacions_dir)

    # Comparativa facturació vs unitats
    print("\n FASE 2.3.3. COMPARATIVA FACTURACIÓ VS UNITATS")
    print("\n Període 2016-2017")
    for cat in top_units_2016_2017['categoria_catala']:
        fact = top_2016_2017[top_2016_2017['categoria_catala'] == ↵
cat]['price'].values

```

```

        units =_
top_units_2016_2017[top_units_2016_2017['categoria_catala']] ==_
cat]['unitats_venudes'].values
        if len(fact) > 0 and len(units) > 0:
            preu_mitja = fact[0] / units[0] if units[0] != 0 else 0
            print(f"- {cat}: {units[0]:,} unitats | BRL{fact[0]/1000:,.2f}K | Preu mig: BRL{preu_mitja:.2f}")

print("\n Període 2017-2018")
for cat in top_units_2017_2018['categoria_catala']:
    fact = top_2017_2018[top_2017_2018['categoria_catala']] ==_
cat]['price'].values
    units =_
top_units_2017_2018[top_units_2017_2018['categoria_catala']] ==_
cat]['unitats_venudes'].values
    if len(fact) > 0 and len(units) > 0:
        preu_mitja = fact[0] / units[0] if units[0] != 0 else 0
        print(f"- {cat}: {units[0]:,} unitats | BRL{fact[0]/1000:,.2f}K | Preu mig: BRL{preu_mitja:.2f}")
except Exception as e:
    print(f" ERROR en comparativa de períodes: {str(e)}")
# -----#
# -----
print("\n" + "-"*50)
print(" FASE 2.4. EVOLUCIÓ ANUAL FACTURACIÓ PER CATEGORIA (2016-2018)")
try:
    min_date = df_final['order_purchase_timestamp'].min().
strftime('%Y-%m')
    max_date = df_final['order_purchase_timestamp'].max().
strftime('%Y-%m')
    print(f" Període de dades: {min_date} a {max_date}")
    df_estacionalitat_anual =_
plot_estacionalitat_anual_per_categoria(df_final)
    print(" Anàlisi d'evolució anual completat per a totes les categories")
    if df_estacionalitat_anual is not None:
        print("\n RESUM ESTADÍSTIC:")
        df_anys = df_estacionalitat_anual.copy()
        df_anys['any'] = df_anys['order_purchase_timestamp'].dt.year
        creixement = []
        for categoria in df_anys['categoria_catala'].unique():
            df_cat = df_anys[df_anys['categoria_catala'] == categoria]
            vendes_2016 = df_cat[df_cat['any'] == 2016]['price'].sum()
            vendes_2017 = df_cat[df_cat['any'] == 2017]['price'].sum()
            vendes_2018 = df_cat[df_cat['any'] == 2018]['price'].sum()

```

```

        if vendes_2016 > 0 and vendes_2018 > 0:
            tasa_creixement = (vendes_2018 - vendes_2016) / vendes_2016
            creixement.append((categoria, tasa_creixement))
    if creixement:
        categoria_max_creix, max_creix = max(creixement, key=lambda x: x[1])
        print(f"- Categoria amb major creixement (2016-2018): {categoria_max_creix} ({max_creix:.1%})")
        mes_max = df_anys.groupby('mes')['price'].sum().idxmax()
        print(f"- Mes amb més vendes: {MESOS_CATALA[mes_max]}")
    except Exception as e:
        print(f" ERROR en anàlisi d'evolució anual: {str(e)}")
# -----
# -----
print("\n" + "-"*50)
print(" FASE 2.5. EVOLUCIÓ ANUAL UNITATS PER CATEGORIA (2016-2018)")
try:
    # Implementació de la funció per unitats
    print(" Generant evolució anual per unitats...")

    # Crear còpia del dataframe per evitar warnings
    df_units = df_final.copy()

    # Agregar dades per unitats
    df_units['any'] = df_units['order_purchase_timestamp'].dt.year
    df_units['mes'] = df_units['order_purchase_timestamp'].dt.month

    # Seleccionar les 10 categories principals
    top_categories = df_units['categoria_catala'].value_counts().nlargest(10).index

    plt.figure(figsize=(16, 12))
    plt.suptitle('Evolució Anual de Unitats Venudes per Categoria TOP 10 Vendes', fontsize=20, y=0.95)

    for i, categoria in enumerate(top_categories, 1):
        df_cat = df_units[df_units['categoria_catala'] == categoria]
        df_agregat = df_cat.groupby(['any', 'mes'])['order_item_id'].count().reset_index()

        ax = plt.subplot(5, 2, i)
        sns.lineplot(
            data=df_agregat,
            x='mes',

```

```

        y='order_item_id',
        hue='any',
        palette='viridis',
        marker='o',
        ax=ax
    )

    ax.set_title(categoria, fontsize=14)
    ax.set_xlabel('')
    ax.set_ylabel('Unitats Venudes')
    ax.set_xticks(range(1, 13))
    ax.set_xticklabels([MESOS_CATALA[m] [:3] for m in range(1, 13)])
    ax.grid(True, alpha=0.2)
    ax.legend(title='Any')

plt.tight_layout(rect=[0, 0, 1, 0.96])
output_path = os.path.join(visualitzacions_dir, "evolucio_anual_unitats.png")
plt.savefig(output_path, dpi=300)
plt.show()
print(f" Gràfic d'evolució anual per unitats guardat a:{output_path}")

except Exception as e:
    print(f" ERROR en anàlisi d'evolució anual per unitats: {str(e)}")
    traceback.print_exc()
# -----


end_analisis_exploratori = time.time()
print(f"\n Temps d'execució de l'anàlisi exploratòria:{end_analisis_exploratori - analisis_start:.2f} segons")
# -----
print("\n" + "-"*50)
print(" FASE 2.6. HEATMAP DE VENDES PER TRAMS HORARIS")
try:
    visualitzacions_dir = os.path.join(BASE_PATH, "visualitzacions")
    os.makedirs(visualitzacions_dir, exist_ok=True)
    heatmap_data = plot_heatmap_vendes_complet(df_final, visualitzacions_dir)

    if heatmap_data is not None:
        print("\n INSIGHTS DEL HEATMAP:")
        max_vendes = heatmap_data.max().max()
        max_idx = heatmap_data.stack().idxmax()
        print(f"• Hora i dia amb més vendes: {max_idx[0]} ({max_idx[1]}) amb {max_vendes:.1f}K BRL")
        min_vendes = heatmap_data.min().min()
        min_idx = heatmap_data.stack().idxmin()

```

```

        print(f"• Hora i dia amb menys vendes: {min_idx[0]}_"
↳({min_idx[1]}) amb {min_vendes:.1f}K BRL")
    except Exception as e:
        print(f" ERROR en generar heatmap: {str(e)}")
    # -----
    # -----
    print("\n" + "-"*50)
    print(" FASE 2.7. HEATMAP D'ESTACIONALITAT PER CATEGORIA")
    try:
        df_estacionalitat = analitzar_estacionalitat_per_categoria(df_final)
        if df_estacionalitat is not None:
            # Generar heatmap general
            visualitzar_estacionalitat(df_estacionalitat)

    except Exception as e:
        print(f" ERROR en heatmap d'estacionalitat: {str(e)}")
        traceback.print_exc()
    # -----
    # -----
    print("\n" + "-"*50)
    print(" FASE 2.8. ANÀLISI DE SATISFACCIÓ DELS CLIENTS")
    try:
        # 1. Relació temps lliurament-satisfacció
        print("\n Generant Boxplot de relació de temps lliurament -_
↳satisfacció...")
        # Verificar que les columnes necessàries existeixen
        if 'review_score' not in df_final.columns or 'delivery_days' not in_
↳df_final.columns:
            raise ValueError("Falten columnes necessàries per a l'anàlisi")

        # Crear còpia del DataFrame i arrodonir les puntuacions
        df_plot = df_final[['review_score', 'delivery_days']].copy()

        # Netejar valors NaN
        df_plot = df_plot.dropna(subset=['review_score', 'delivery_days'])

        # Convertir puntuacions a enters
        df_plot['review_score'] = df_plot['review_score'].round(0).
↳astype(int)

        # Filtrar puntuacions vàlides

```

```

df_plot = df_plot[(df_plot['review_score'] >= 1) &
                   (df_plot['review_score'] <= 5)]

# Verificar si hi ha dades suficients
if df_plot.empty:
    print(" AVÍS: No hi ha dades suficients per generar el gràfic")
else:
    # Configurar l'estil de Seaborn
    sns.set(style="whitegrid")

    plt.figure(figsize=(10, 6))
    ax = sns.boxplot(
        data=df_plot,
        x='review_score',
        y='delivery_days',
        showfliers=False,
        palette='Blues',
        order=sorted(df_plot['review_score'].unique())
    )

    # Afegir recompte de dades a cada boxplot
    counts = df_plot.groupby('review_score').size()
    for i, score in enumerate(sorted(df_plot['review_score'].
                                         unique())):
        count = counts.get(score, 0)
        ax.text(i, ax.get_ylim()[0] - 2, f'n={count}',
                ha='center', va='top', fontsize=9)

    plt.title('Temps de Lliurament per Nivell de Satisfacció', u
               fontsize=16)
    plt.xlabel('Puntuació (arrodonida a enter)', fontsize=12)
    plt.ylabel('Dies de Lliurament', fontsize=12)
    plt.xticks(fontsize=10)
    plt.yticks(fontsize=10)
    plt.tight_layout()
    plt.savefig(os.path.join(visualitzacions_dir, u
                           "temps_satisfaccio.png"), dpi=300)
    plt.show()
    print(" Boxplot generat correctament")

except Exception as e:
    print(f" ERROR en generar el boxplot: {str(e)}")
    traceback.print_exc()

# 2. Matriu de correlació millorada
try:
    print("\n Generant matriu de correlació millorada...")

```

```

resultats_correlacio = matriu_correlacio_millorada(
    df=df_final,
    visualitzacions_dir=visualitzacions_dir,
    cols=['review_score', 'delivery_days', 'price_normalized'],
    filename="correlacions_satisfacció_millorat.png"
)
if resultats_correlacio is not None:
    print(" Matriu de correlació generada correctament")
else:
    print(" No s'ha pogut generar la matriu de correlació")

except Exception as e:
    print(f" ERROR en generar la matriu de correlació: {str(e)}")
    traceback.print_exc()

# 3. Matriu de satisfacció per mes i categoria
try:
    print("\n Generant matriu de satisfacció per mes i categoria...")

    # Agregar dades
    satisfacció_mes = df_final.groupby(['categoria_catala', 'mes'])['review_score'].mean().unstack()

    # Seleccionar les 10 categories més importants
    top_categories = df_final['categoria_catala'].value_counts().nlargest(10).index

    # Filtrar només les categories top
    satisfacció_mes = satisfacció_mes.loc[satisfacció_mes.index.isin(top_categories)]

    # Completar mesos faltants amb NaN
    all_months = list(range(1, 13))
    for month in all_months:
        if month not in satisfacció_mes.columns:
            satisfacció_mes[month] = np.nan

    # Ordenar columnes per mes
    satisfacció_mes = satisfacció_mes[all_months]

    # Configurar etiquetes dels mesos
    mesos_labels = [MESOS_CATALA.get(m, f'M{m}') for m in all_months]

    plt.figure(figsize=(14, 10))
    sns.heatmap(
        satisfacció_mes,
        annot=True,

```

```

        fmt=".1f",
        cmap="YlGnBu",
        linewidths=0.5,
        cbar_kws={'label': 'Satisfacció Mitjana'},
        vmin=1, # Valor mínim possible
        vmax=5 # Valor màxim possible
    )

    plt.title('Satisfacció Mitjana per Mes i Categoria (Top 10 Categories)', fontsize=18)
    plt.xlabel('Mes')
    plt.ylabel('Categoria')
    plt.xticks(ticks=np.arange(len(mesos_labels)),
               labels=mesos_labels,
               rotation=45)
    plt.tight_layout()
    output_path = os.path.join(visualitzacions_dir, "matriu_satisfaccio_mes_categoria.png")
    plt.savefig(output_path, dpi=300)
    plt.show()
    print(f" Gràfic guardat a: {output_path}")

    # Informació addicional per a depuració
    print(f" Estadístiques de la matriu:")
    print(f"- Categories incloses: {len(satisfaccio_mes)}")
    print(f"- Mesos inclosos: {len(satisfaccio_mes.columns)}")
    print(f"- Valors calculats: {satisfaccio_mes.size}")
    print(f"- Valors NaN: {satisfaccio_mes.isna().sum().sum()}")

except Exception as e:
    print(f" ERROR en generar la matriu de satisfacció: {str(e)}")
    traceback.print_exc()

# -----
end_analisis_exploratori = time.time()
print(f"\n Temps d'execució de l'anàlisi exploratòria: {end_analisis_exploratori - analisis_start:.2f} segons")

# -----
print("\n" + "="*100)
print(" FASE 3: ANÀLISI ESTRATÈGIC")
fase3_start = time.time()

# -----
print("\n" + "-"*50)
print(" FASE 3.2. ANÀLISI DE PREU I DEMANDA")
try:

```

```

        df_preu_demanda = analisi_preu_demanda(df_final)
    except Exception as e:
        print(f" ERROR en anàlisi de preu i demanda: {str(e)}")
    # ----

    # -----
    print("\n" + "-"*50)
    print(" FASE 3.3. IMPACTE DE FOTOS EN PRODUCTES")
    try:
        analitzar_impacte_fotos_producte(df_final)
    except Exception as e:
        print(f" ERROR en anàlisi d'impacte de fotos: {str(e)}")
    # ----

    end_fase3 = time.time()
    print(f"\n Temps d'execució de l'anàlisi estratègica: {end_fase3 - fase3_start:.2f} segons")

    # -----
    print("\n" + "="*100)
    print(" FASE 4: MODELATGE PREDICTIU")
    fase4_start = time.time()

    # -----
    print("\n" + "-"*50)
    print(" FASE 4.1. PREPARACIÓ DE DADES PER MODELATGE")
    try:
        df_model = preparar_dades_per_model(df_final)
    except Exception as e:
        print(f" ERROR en preparació de dades per modelatge: {str(e)}")
        df_model = None
    # ----

    # -----
    print("\n" + "-"*50)
    print(" FASE 4.2. ENTENAMENT DE MODELS PER CATEGORIA")
    try:
        if df_model is not None:
            print("\n ENTENANT MODELS PER A 15 CATEGORIES PRINCIPALS")
            resultats_models = entrenar_models_per_totes_categories(df_model, top_n=15)
        else:
            resultats_models = {}
            print(" No es poden entrenar models sense dades preparades")
    except Exception as e:
        print(f" ERROR en entrenament de models: {str(e)}")
        resultats_models = {}

```

```

        traceback.print_exc()
# -----
#
# -----
print("\n" + "-"*50)
print(" FASE 4.3. AVALUACIÓ I COMPARATIVA DE MODELS")
try:
    if resultats_models:
        print("\n RESULTATS DELS MODELS")

        # 1. Generar ranking de top5 i pitjor5 models
        top5, pitjor5 = visualitzar_top_pitjor_models(resultats_models)

        # 2. Generar comparativa detallada
        df_comparatiu = comparar_top_pitjor_models(resultats_models, ↴
                                                    top5, pitjor5)

        # 3. Generar mètriques i comparacions
        visualitzar_metrics_top_categories(resultats_models)
        visualitzar_correlacio_metrics(resultats_models)
        comparar_models_en_categories(resultats_models)
        generar_resum_comparatiu(resultats_models)
    else:
        print(" No hi ha resultats per avaluar")
except Exception as e:
    print(f" ERROR en evaluació de models: {str(e)}")
    traceback.print_exc()
# -----
#
# -----
print("\n" + "-"*50)
print(" FASE 4.4. VISUALITZACIÓ DE RESULTATS")
try:
    if resultats_models:
        # 1. Generar ranking de top5 i pitjor5 models (si no s'ha fet a ↴
        # l'apartat anterior)
        if 'top5' not in locals():
            top5, pitjor5 = ↴
            visualitzar_top_pitjor_models(resultats_models)

            print("\n VISUALITZANT PREDICCIIONS")
            mostrar_predicccions_top_models(resultats_models, top5)
            mostrar_predicccions_pitjor_models(resultats_models, pitjor5)
        else:
            print(" No hi ha resultats per visualitzar")
    except Exception as e:
        print(f" ERROR en visualització de resultats: {str(e)}")

```

```

        traceback.print_exc()
# -----
```

```

    end_fase4 = time.time()
    print(f"\n Temps d'execució del modelatge predictiu: {end_fase4 -_
↳fase4_start:.2f} segons")

# -----
print("\n" + "="*100)
print(" VERIFICACIÓ FINAL DE RESULTATS")
try:
    # VERIFICACIÓ FINAL DE VISUALITZACIONS
    print("\n" + "="*50)
    print(" VERIFICACIÓ DE VISUALITZACIONS GENERADES")
```

```

    visualitzacions_generades = {
        'comparativa_facturacio_periodes': False,
        'comparativa_unitats_periodes': False,
        'heatmap_estacionalitat_categories': False,
        'comparacio_metrics_models': False,
        'correlacio_metrics': False
    }

# Comprovar existència d'arxius
visualitzacions_dir = os.path.join(BASE_PATH, "visualitzacions")
prediccions_dir = os.path.join(BASE_PATH, "prediccions")

# Verificar gràfics de facturació
if (os.path.exists(os.path.join(visualitzacions_dir,_
↳"comparativa_facturacio_periodes.png"))):
    visualitzacions_generades['comparativa_facturacio_periodes'] =_
↳True

# Verificar gràfics d'unitats
if (os.path.exists(os.path.join(visualitzacions_dir,_
↳"comparativa_unitats_periodes.png"))):
    visualitzacions_generades['comparativa_unitats_periodes'] = True

# Verificar heatmap
if os.path.exists(os.path.join(visualitzacions_dir,_
↳"estacionalitat_general.png")):
    visualitzacions_generades['heatmap_estacionalitat_categories'] =_
↳= True
```

```

# Verificar comparació de mètriques
```

```

        if os.path.exists(os.path.join(prediccions_dir, "comparacio_models_categories.png")):
            visualitzacions_generades['comparacio_metrics_models'] = True

    # Verificar correlació de mètriques
    if os.path.exists(os.path.join(prediccions_dir, "correlacio_metrics.png")):
        visualitzacions_generades['correlacio_metrics'] = True

    # Informe final
    print(" ESTAT DE VISUALITZACIONS:")
    for viz, status in visualitzacions_generades.items():
        print(f"- {viz}: {'GENERADA' if status else 'NO GENERADA'}")
    except Exception as e:
        print(f" ERROR en verificació final: {str(e)}")
    # -----
    print("\n" + "="*100)
    print(f" TOTES LES ETAPES COMPLETADES. Temps total: {time.time() - start_total:.2f} segons")
    print("="*100)

except Exception as e:
    print(f"\n ERROR NO CONTROLAT: {str(e)}")
    traceback.print_exc()

```

=====

=====

EXECUCIÓ PRINCIPAL: PREPROCESSAMENT I ANÀLISI

=====

=====

=====

=====

FASE 1: PREPROCESSAMENT DE DADES

Carregant olist_orders_dataset.csv...

olist_orders_dataset.csv carregat i timestamps convertits

Carregant olist_customers_dataset.csv...

olist_customers_dataset.csv carregat i timestamps convertits

Carregant olist_order_items_dataset.csv...

olist_order_items_dataset.csv carregat i timestamps convertits

Carregant olist_products_dataset.csv...

olist_products_dataset.csv carregat i timestamps convertits

Carregant product_category_name_translation.csv...

product_category_name_translation.csv carregat i timestamps convertits

Carregant olist_order_reviews_dataset.csv...

olist_order_reviews_dataset.csv carregat i timestamps convertits

```

Carregant olist_sellers_dataset.csv...
olist_sellers_dataset.csv carregat i timestamps convertits
Carregant olist_order_payments_dataset.csv...
olist_order_payments_dataset.csv carregat i timestamps convertits
=====
```

===== VERIFICANT RELACIONS I QUALITAT DE DADES

```

=====  
VERIFICACIÓ DE RELACIONS 1:N I QUALITAT DE DADES
Relació 1:1 verificada: Ordres úniques
Avís: S'han detectat relacions 1:N en pagaments per ordre
- Màxim pagaments per ordre: 29
- Mínim pagaments per ordre: 1
- Mitjana pagaments per ordre: 1.04
```

INFORME DE QUALITAT PER TAULA:

TAULA: ORDERS

	missing %	dtype	unique_values \
order_delivered_customer_date	2.982	datetime64[ns]	95664
order_delivered_carrier_date	1.793	datetime64[ns]	81018
order_approved_at	0.161	datetime64[ns]	90733
order_id	0.000	object	99441
order_purchase_timestamp	0.000	datetime64[ns]	98875
order_status	0.000	object	8
customer_id	0.000	object	99441
order_estimated_delivery_date	0.000	datetime64[ns]	459
			exemple_valor
order_delivered_customer_date		2017-10-10 21:25:13	
order_delivered_carrier_date		2017-10-04 19:55:00	
order_approved_at		2017-10-02 11:07:15	
order_id		e481f51cbdc54678b7cc49136f2d6af7	
order_purchase_timestamp		2017-10-02 10:56:33	
order_status		delivered	
customer_id		9ef432eb6251297304e76186b10a928d	
order_estimated_delivery_date		2017-10-18 00:00:00	

TAULA: CUSTOMERS

	missing %	dtype	unique_values \
customer_id	0.000	object	99441
customer_unique_id	0.000	object	96096
customer_zip_code_prefix	0.000	int64	14994
customer_city	0.000	object	4119
customer_state	0.000	object	27

	exemple_valor
customer_id	06b8999e2fba1a1fbc88172c00ba8bc7
customer_unique_id	861eff4711a542e4b93843c6dd7febb0
customer_zip_code_prefix	14409
customer_city	franca
customer_state	SP

TAULA: ORDER_ITEMS

	missing_%	dtype	unique_values	\
order_id	0.000	object	98666	
order_item_id	0.000	int64	21	
product_id	0.000	object	32951	
seller_id	0.000	object	3095	
shipping_limit_date	0.000	datetime64[ns]	93318	
price	0.000	float64	5968	
freight_value	0.000	float64	6999	

	exemple_valor
order_id	00010242fe8c5a6d1ba2dd792cb16214
order_item_id	1
product_id	4244733e06e7ecb4970a6e2683c13e61
seller_id	48436dade18ac8b2bce089ec2a041202
shipping_limit_date	2017-09-19 09:45:35
price	58.900
freight_value	13.290

TAULA: PRODUCTS

	missing_%	dtype	unique_values	\
product_category_name	1.851	object	73	
product_description_lenght	1.851	float64	2960	
product_name_lenght	1.851	float64	66	
product_photos_qty	1.851	float64	19	
product_weight_g	0.006	float64	2204	
product_height_cm	0.006	float64	102	
product_length_cm	0.006	float64	99	
product_width_cm	0.006	float64	95	
product_id	0.000	object	32951	

	exemple_valor
product_category_name	perfumaria
product_description_lenght	287.000
product_name_lenght	40.000
product_photos_qty	1.000
product_weight_g	225.000
product_height_cm	10.000

product_length_cm	16.000
product_width_cm	14.000
product_id	1e9e8ef04dbcff4541ed26657ea517e5

TAULA: TRADUCCIO

	missing_%	dtype	unique_values	exemple_valor
product_category_name	0.000	object	71	beleza_saude
product_category_name_english	0.000	object	71	health_beauty

TAULA: REVIEWS

	missing_%	dtype	unique_values	\
review_comment_title	88.342	object	4527	
review_comment_message	58.703	object	36159	
review_id	0.000	object	98410	
review_score	0.000	int64	5	
order_id	0.000	object	98673	
review_creation_date	0.000	datetime64[ns]	636	
review_answer_timestamp	0.000	datetime64[ns]	98248	

	exemple_valor
review_comment_title	NaN
review_comment_message	NaN
review_id	7bc2406110b926393aa56f80a40eba40
review_score	4
order_id	73fc7af87114b39712e6da79b0a377eb
review_creation_date	2018-01-18 00:00:00
review_answer_timestamp	2018-01-18 21:46:59

TAULA: SELLERS

	missing_%	dtype	unique_values	\
seller_id	0.000	object	3095	
seller_zip_code_prefix	0.000	int64	2246	
seller_city	0.000	object	611	
seller_state	0.000	object	23	

	exemple_valor
seller_id	3442f8959a84dea7ee197c632cb2df15
seller_zip_code_prefix	13023
seller_city	campinas
seller_state	SP

TAULA: PAYMENTS

	missing_%	dtype	unique_values	\
order_id	0.000	object	99440	

```
payment_sequential      0.000    int64          29
payment_type            0.000    object           5
payment_installments   0.000    int64          24
payment_value           0.000    float64        29077
```

```
exempla_valor
order_id                b81ef226f3fe1789b1e8b2acac839d17
payment_sequential       1
payment_type             credit_card
payment_installments    8
payment_value            99.330
```

Processant categories de productes...
S'han eliminat 623 productes sense traducció a l'anglès
Productes sense traducció després del neteja: 0

Fusionant taules per crear el dataset final...
Memòria alliberada de datasets intermedis

INVESTIGANT PREUS INVÀLIDS (0 O NEGATIUS)
No s'han detectat preus invàlids

GESTIÓ DE VALORS NULS EN TIMESTAMP

Total de registres a processar: 113,425
order_approved_at: 161 valors imputats amb order_purchase_timestamp
order_delivered_carrier_date: 1968 (1.74%) valors imputats
order_delivered_customer_date: 3229 (2.85%) valors imputats
Valors nuls en timestamps gestionats

NORMALITZACIÓ DE PREUS PER CATEGORIA

Normalitzant preus per categories...
Preus normalitzats per a 68 categories
Preus normalitzats per categoria

OPTIMITZACIÓ DE MEMÒRIA

Categories catalanes optimitzades. Cardinalitat: 52
product_photos_qty: Valors NaN imputats. Nuls restants: 0
product_weight_g: 2403 (2.12%) valors NaN imputats a 0
product_length_cm: 2403 (2.12%) valors NaN imputats a 0
product_height_cm: 2403 (2.12%) valors NaN imputats a 0
product_width_cm: 2403 (2.12%) valors NaN imputats a 0
Memòria optimitzada

```
=====
AFEGINT VARIABLES TEMPORALS
Afegint variables temporals...
S'han afegit 5 noves variables comercials brasileres
Variables temporals totals: 19
Variable 'delivery_days' afegida correctament
Variables temporals afegides
```

```
=====
DATAFRAME FINAL PREPARAT
Registres: 113,425
Columnes: 48
Ús de memòria: 87.96 MB
```

```
=====
Dataset exportat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT
10\Brazilian E-Commerce Public
Dataset\preprocessat\ecommerce_brasil_preprocessat.parquet
```

===== INFORME FINAL DE QUALITAT DE DADES

IMPUTACIONS DE TEMPS:

IMPUTACIONS DE PRODUCTES:

- product_weight_g: 2411 (2.13%) valors imputats a 0
- product_length_cm: 2403 (2.12%) valors imputats a 0
- product_height_cm: 2403 (2.12%) valors imputats a 0
- product_width_cm: 2403 (2.12%) valors imputats a 0

RESUM DE VARIABLES CLAU:

- Categories de productes: 52 categories
- Venedors únics: 3095
- Clients únics: 99441
- Interval temporal: De 2016-09-04 21:15:19 a 2018-10-17 17:30:18

VARIABLES D'ESTACIONALITAT:

- Dies festius: 1717 comandes (1.5%)
- Vacances d'estiu: 27045 comandes (23.8%)
- Carnaval: 1435 comandes (1.3%)

DISTRIBUCIÓ DE CATEGORIES CATALANES:

- llit_bany_taula: 11115 comandes (9.8%)
- salut_i_bellesa: 9670 comandes (8.5%)
- esports_oci: 8641 comandes (7.6%)
- mobles_decoració: 8334 comandes (7.3%)
- accessoris_informàtics: 7827 comandes (6.9%)

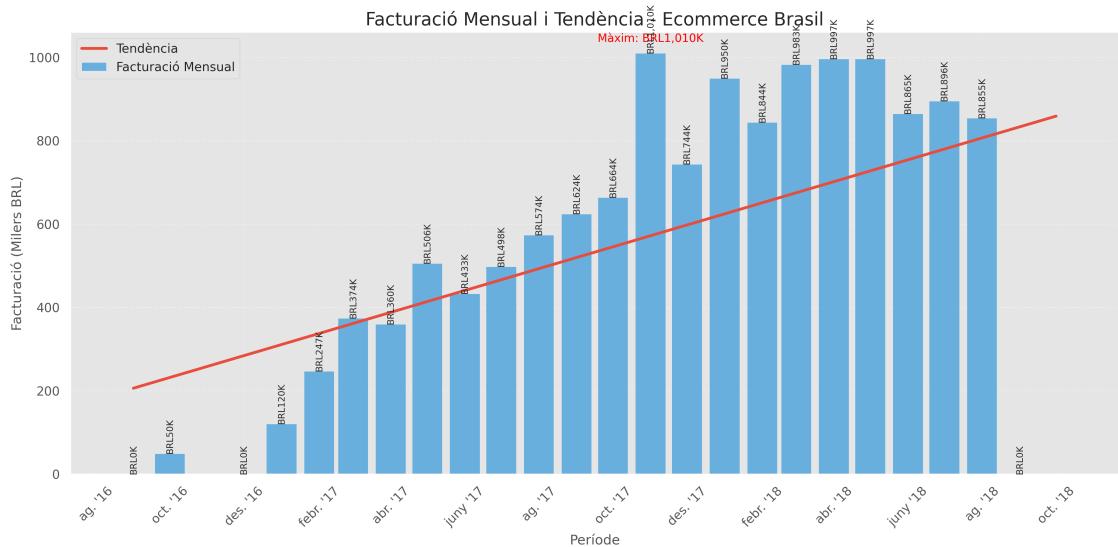
- articles_llar: 6964 comandes (6.1%)
 - rellotges_regals: 5991 comandes (5.3%)
 - telefonia: 4545 comandes (4.0%)
 - eines_jardí: 4347 comandes (3.8%)
 - automòbil: 4235 comandes (3.7%)
-

PROCÉS DE PREPROCESSAMENT COMPLETAT AMB ÈXIT

Temps d'execució del preprocessament: 4.16 segons

FASE 2: ANÀLISI EXPLORATÒRIA

FASE 2.1. ANÀLISI DE FACTURACIÓ MENSUAL EN BRL BRASIL



CONCLUSIONS DE FACTURACIÓ MENSUAL

- Període analitzat: set. '16 a oct. '18
- Facturació total: BRL13,592K
- Mes amb mínima facturació: oct. 18 (BRL0K)
- Mes amb màxima facturació: nov. 17 (BRL1,010K)
- Creixement del període: -100.0%
- Tendència mensual: 0.9K BRL/mes (creixement)

RECOMANACIÓ: El pic de vendes coincideix amb el Black Friday.

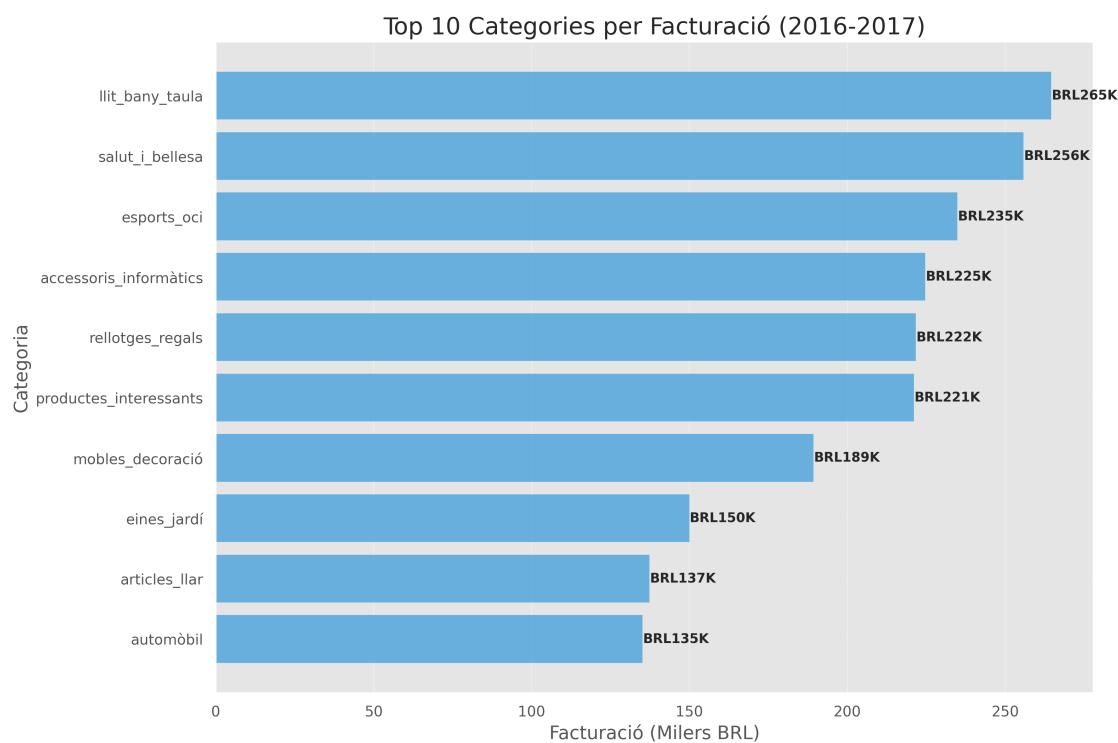
- Optimitzi campanyes de marketing per a aquest període.
 - Asseguri stock suficient per a la demanda.
-

Facturació mensual analitzada. Registres: 25 mesos

FASE 2.2. TOP CATEGORIES PER FACTURACIÓ

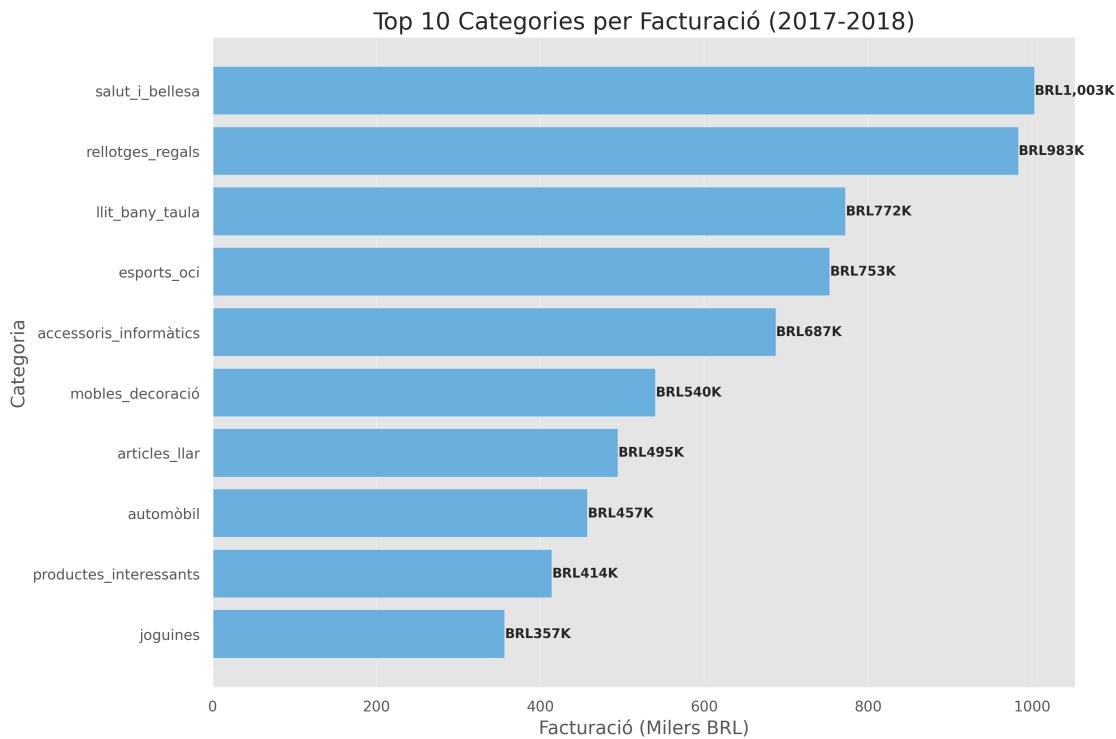
FASE 2.2.1. TOP CATEGORIES PER FACTURACIÓ 2016-2017

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\visualitzacions\top10_categories_2016_2017.png



FASE 2.2.2. TOP CATEGORIES PER FACTURACIÓ 2017-2018

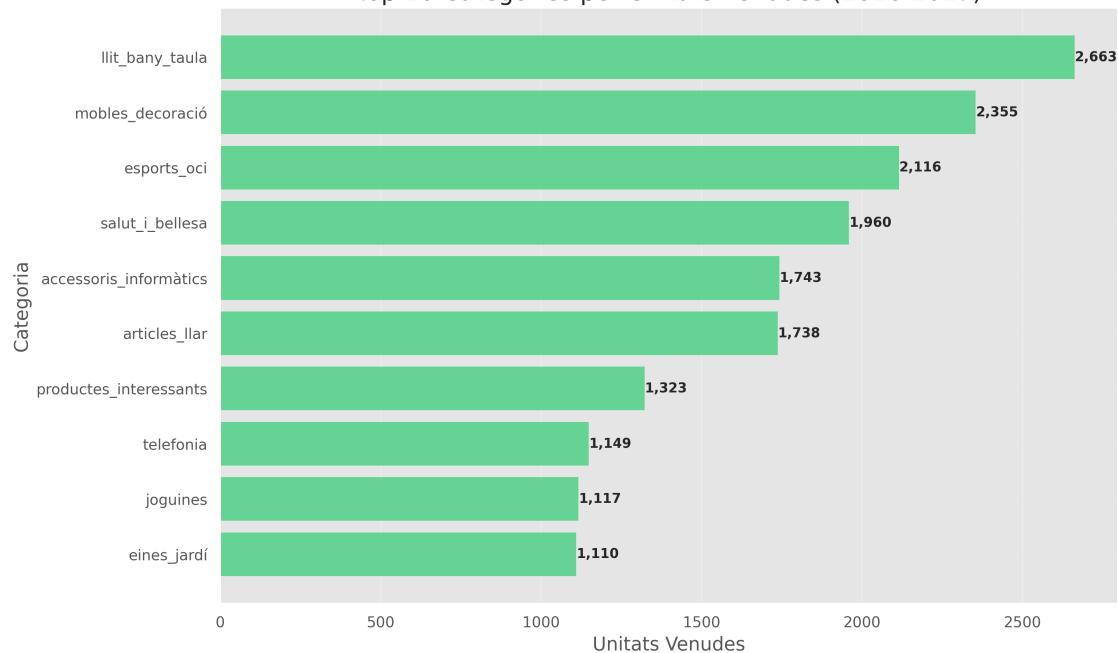
Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\visualitzacions\top10_categories_2017_2018.png



FASE 2.4. TOP CATEGORIES PER UNITATS

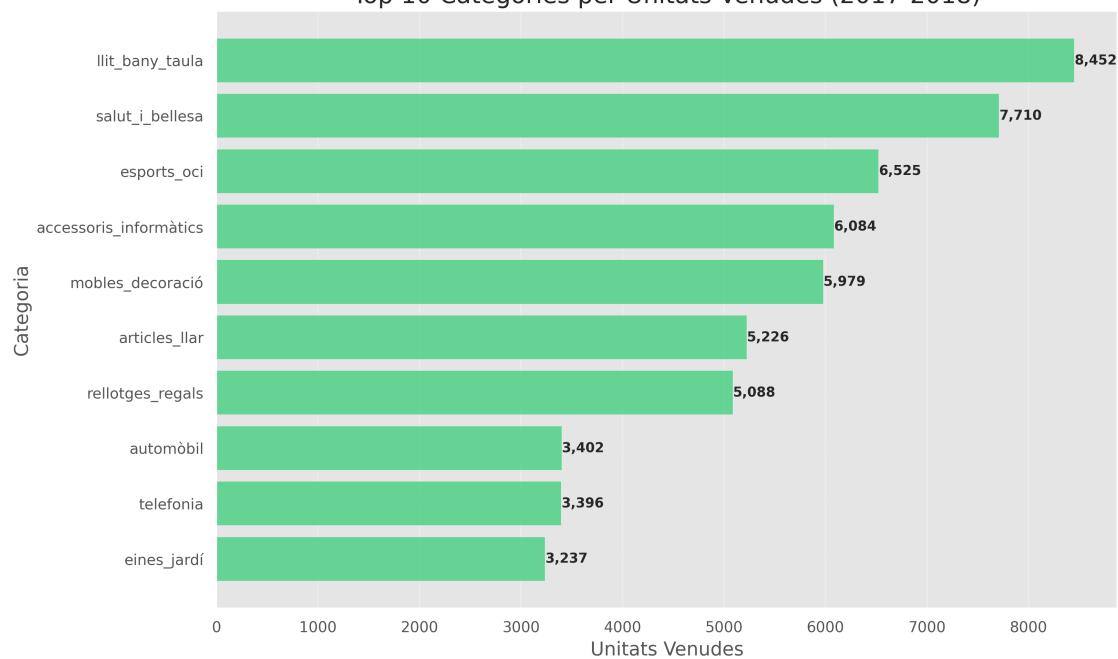
FASE 2.4.1. TOP CATEGORIES PER UNITATS 2016-2017

Top 10 Categories per Unitats Venudes (2016-2017)



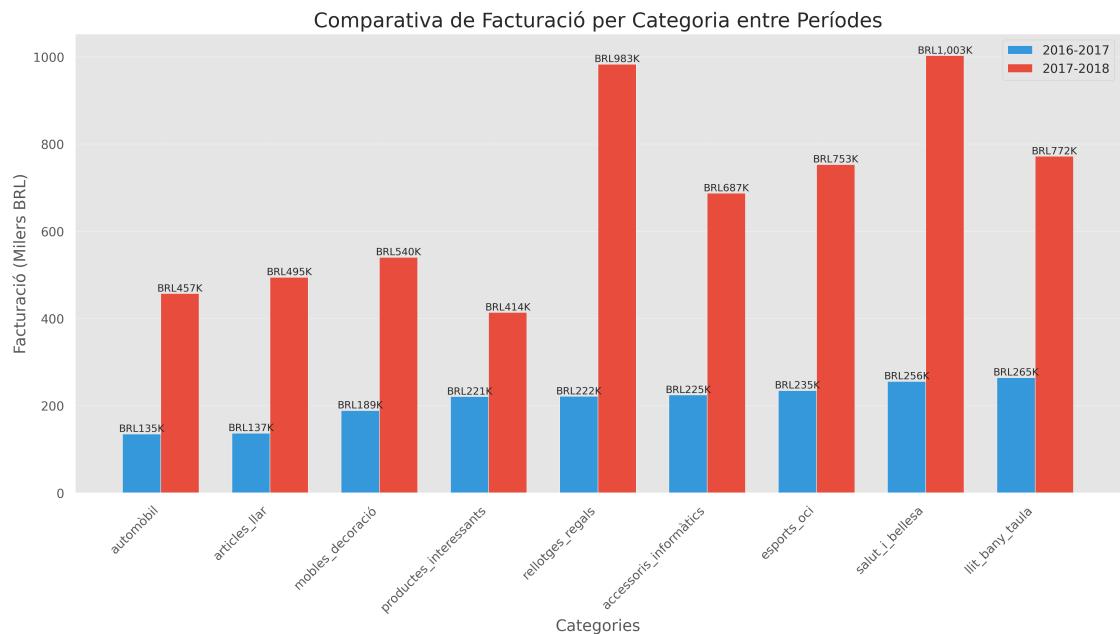
FASE 2.4.2. TOP CATEGORIES PER UNITATS 2017-2018

Top 10 Categories per Unitats Venudes (2017-2018)



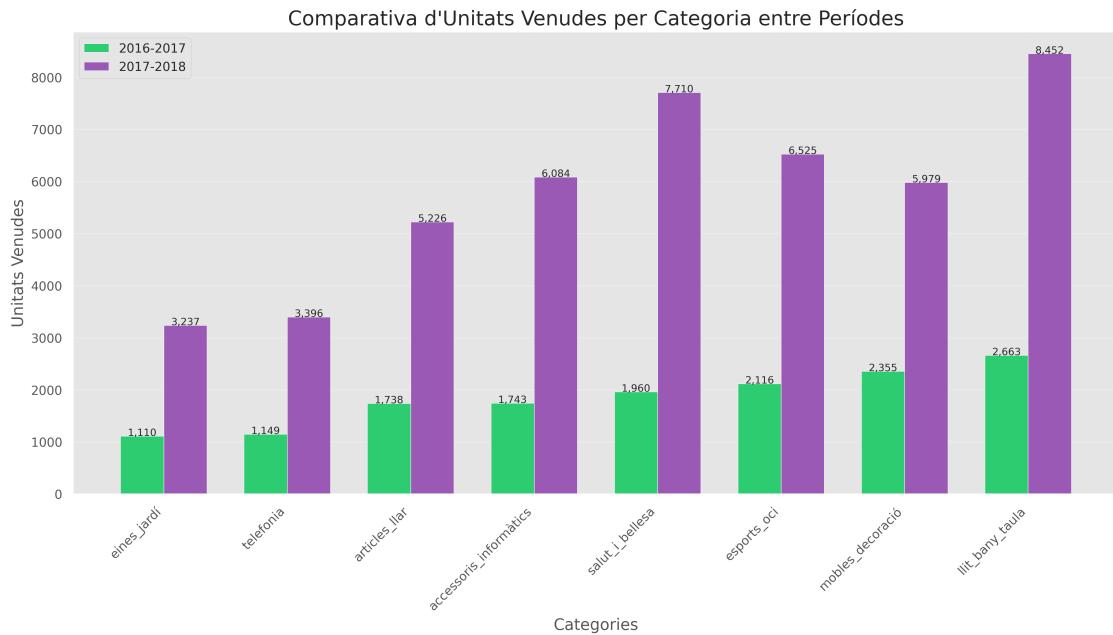
FASE 2.3. COMPARATIVA PERÍODES COMERCIALS

FASE 2.3.1. COMPARATIVA FACTURACIÓ PERÍODES COMERCIALS



Comparativa de facturació guardada a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\visualitzacions\comparativa_facturacio_períodes.png

FASE 2.3.2. COMPARATIVA UNITATS PERÍODES COMERCIALS



Comparativa d'unitats guardada a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\visualitzacions\comparativa_unitats_periodes.png

FASE 2.3.3. COMPARATIVA FACTURACIÓ VS UNITATS

Període 2016-2017

- unes_jardí: 1,110 unitats | BRL150K | Preu mig: BRL135.16
- productes_interessants: 1,323 unitats | BRL221K | Preu mig: BRL167.15
- articles_llar: 1,738 unitats | BRL137K | Preu mig: BRL79.05
- accessoris_informàtics: 1,743 unitats | BRL225K | Preu mig: BRL128.93
- salut_i_bellesa: 1,960 unitats | BRL256K | Preu mig: BRL130.53
- esports_oci: 2,116 unitats | BRL235K | Preu mig: BRL111.03
- mobles_decoració: 2,355 unitats | BRL189K | Preu mig: BRL80.40
- llit_bany_taula: 2,663 unitats | BRL265K | Preu mig: BRL99.39

Període 2017-2018

- automòbil: 3,402 unitats | BRL457K | Preu mig: BRL134.47
- rellotges_regals: 5,088 unitats | BRL983K | Preu mig: BRL193.25
- articles_llar: 5,226 unitats | BRL495K | Preu mig: BRL94.69
- mobles_decoració: 5,979 unitats | BRL540K | Preu mig: BRL90.39
- accessoris_informàtics: 6,084 unitats | BRL687K | Preu mig: BRL112.96
- esports_oci: 6,525 unitats | BRL753K | Preu mig: BRL115.42
- salut_i_bellesa: 7,710 unitats | BRL1,003K | Preu mig: BRL130.07
- llit_bany_taula: 8,452 unitats | BRL772K | Preu mig: BRL91.38

FASE 2.4. EVOLUCIÓ ANUAL FACTURACIÓ PER CATEGORIA (2016-2018)

Període de dades: 2016-09 a 2018-10

Generant estacionalitat anual per categoria...

Anys disponibles: [2016, 2017, 2018]

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT
10\Brazilian E-Commerce Public

Dataset\visualitzacions\estacionalitat_anual_per_categoria.png

Temps de generació: 18.92 segons

Estacionalitat vendes anual per categoria (2016-2018)



Note: Les vendes estan normalitzades dins de cada any (0 = mínim d'any, 1 = màxim d'any) per comparar patrons d'estacionalitat.

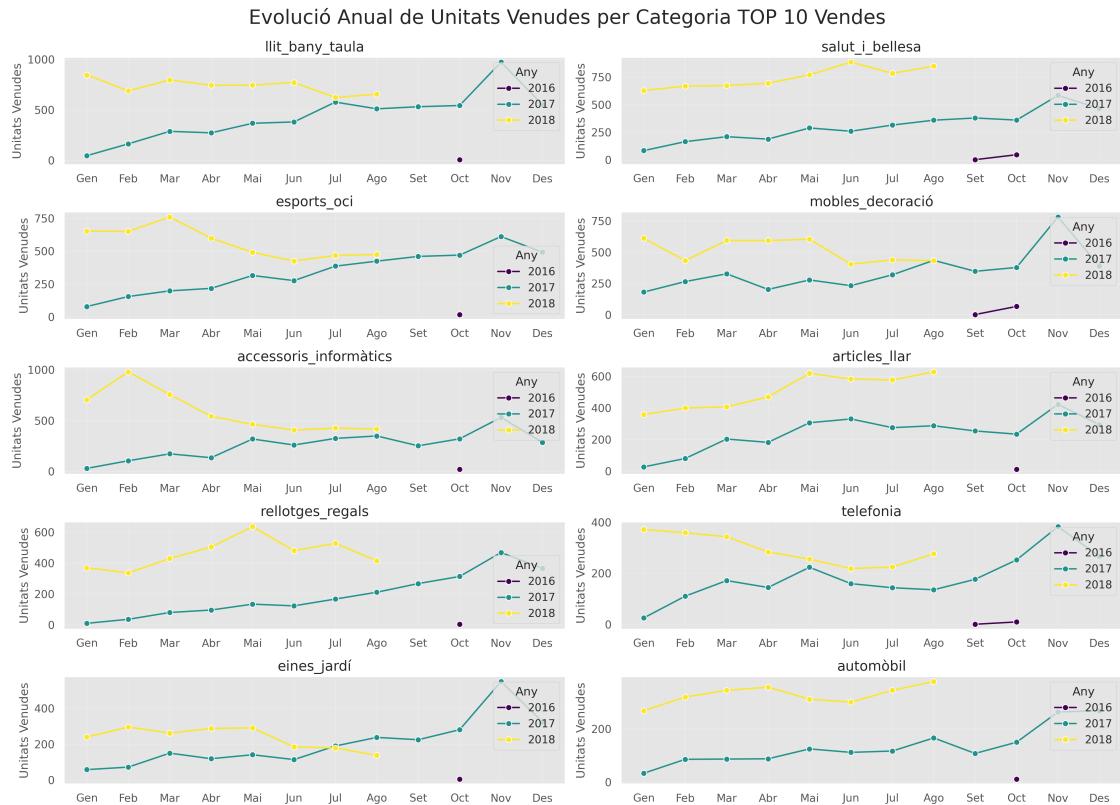
Anàlisi d'evolució anual completat per a totes les categories

RESUM ESTADÍSTIC:

- Categoria amb major creixement (2016-2018): llit_bany_taula (112234.1%)
- Mes amb més vendes: Maig

FASE 2.5. EVOLUCIÓ ANUAL UNITATS PER CATEGORIA (2016-2018)

Generant evolució anual per unitats...



Gràfic d'evolució anual per unitats guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\visualitzacions\evolucion_anual_unitats.png

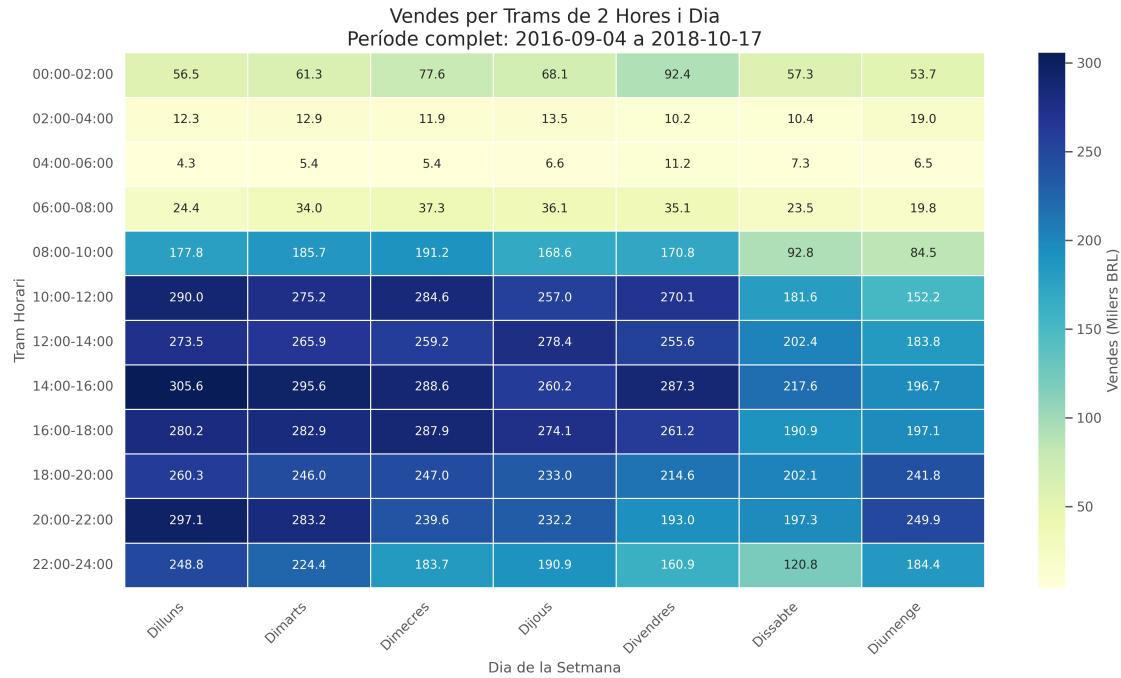
Temps d'execució de l'anàlisi exploratòria: 36.45 segons

FASE 2.6. HEATMAP DE VENDES PER TRAMS HORARIS

Generant mapa de calor de vendes per trams horaris (periode complet)...

Mapa de calor guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data

Analytics\SPRINT 10\Brazilian E-Commerce Public
 Dataset\visualitzacions\heatmap_vendes_periode_complet.png



INSIGHTS DEL HEATMAP:

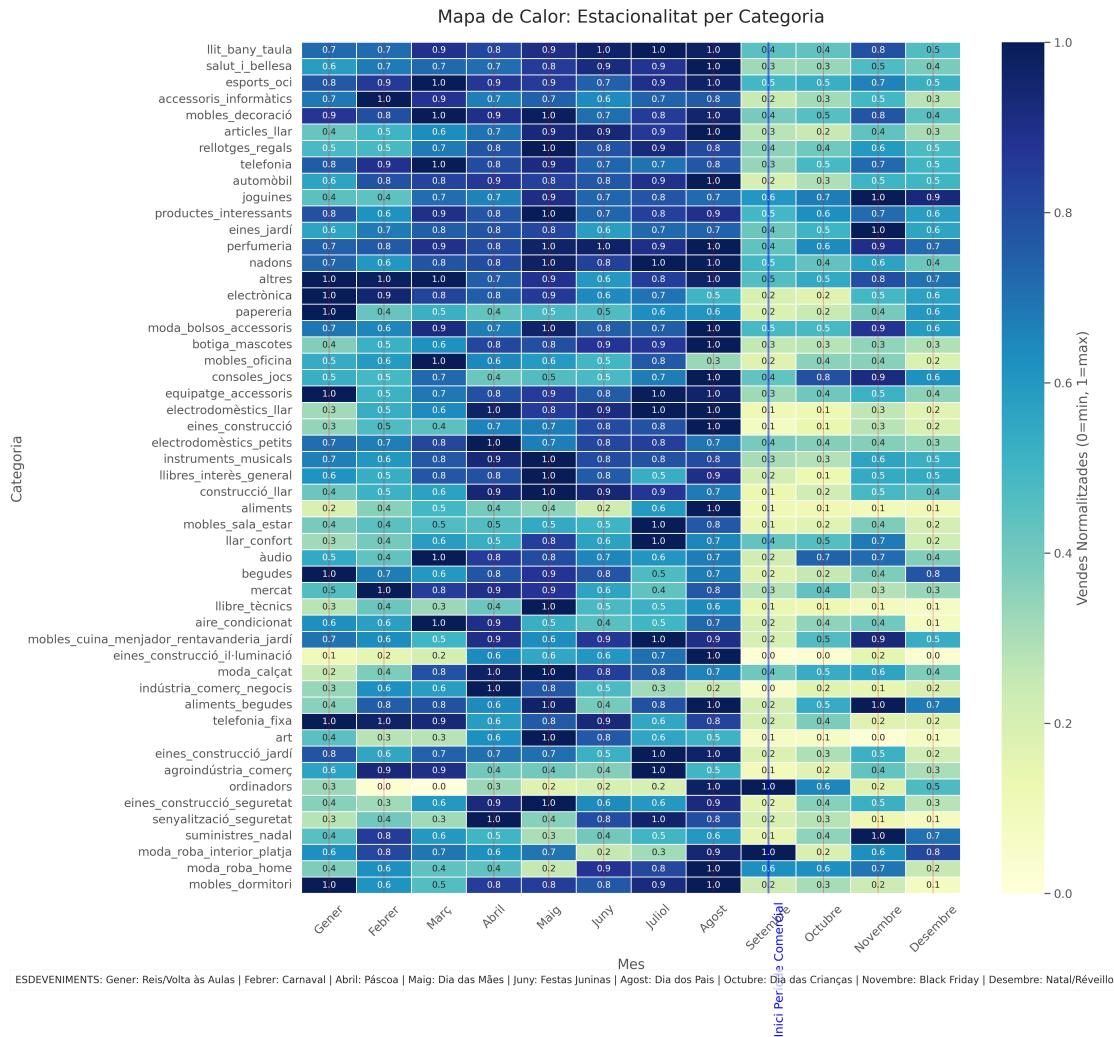
- Hora i dia amb més vendes: 14:00-16:00 (Dilluns) amb 305.6K BRL
- Hora i dia amb menys vendes: 04:00-06:00 (Dilluns) amb 4.3K BRL

FASE 2.7. HEATMAP D'ESTACIONALITAT PER CATEGORIA

Realitzant anàlisi d'estacionalitat per categoria...

Anàlisi d'estacionalitat completat

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public
 Dataset\visualitzacions\estacionalitat_general.png



CONCLUSIONS D'ESTACIONALITAT:

CATEGORIES AMB MAJOR ESTACIONALITAT:

- ordinadors: Variació 1.0 (Pic a Setembre)
- eines_construcció_il·luminació: Variació 1.0 (Pic a Agost)
- art: Variació 1.0 (Pic a Maig)
- indústria_comerç_negocis: Variació 1.0 (Pic a Abril)
- llibre_tècnics: Variació 0.9 (Pic a Maig)

CATEGORIES MÉS ESTABLES:

- productes_interessants: Desviació 0.15
- eines_jardí: Desviació 0.16
- perfumeria: Desviació 0.16
- moda_bolsos_accessoris: Desviació 0.17
- joguines: Desviació 0.17

MESOS AMB MÉS DEMANDA GLOBAL:

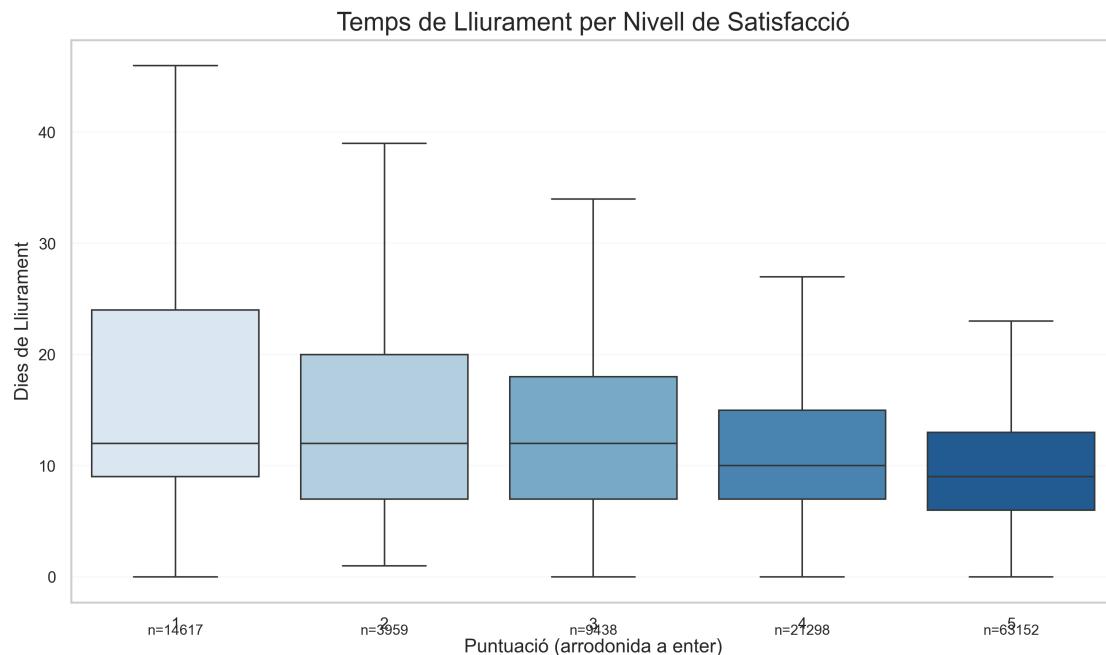
- Agost: 0.8 (sobre mitjana)
- Maig: 0.8 (sobre mitjana)
- Juliol: 0.8 (sobre mitjana)

IMPACTE D'ESDEVENIMENTS ESPECÍFICS:

- Reis/Volta às Aulas (Gener): 0.6
- Carnaval (Febrer): 0.6
- Páscoa (Abril): 0.7
- Dia das Mães (Maig): 0.8
- Festas Juninas (Juny): 0.7
- Dia dos Pais (Agost): 0.8
- Dia das Crianças (Octubre): 0.4
- Black Friday (Novembre): 0.5
- Natal/Réveillon (Desembre): 0.4

FASE 2.8. ANÀLISI DE SATISFACCIÓ DELS CLIENTS

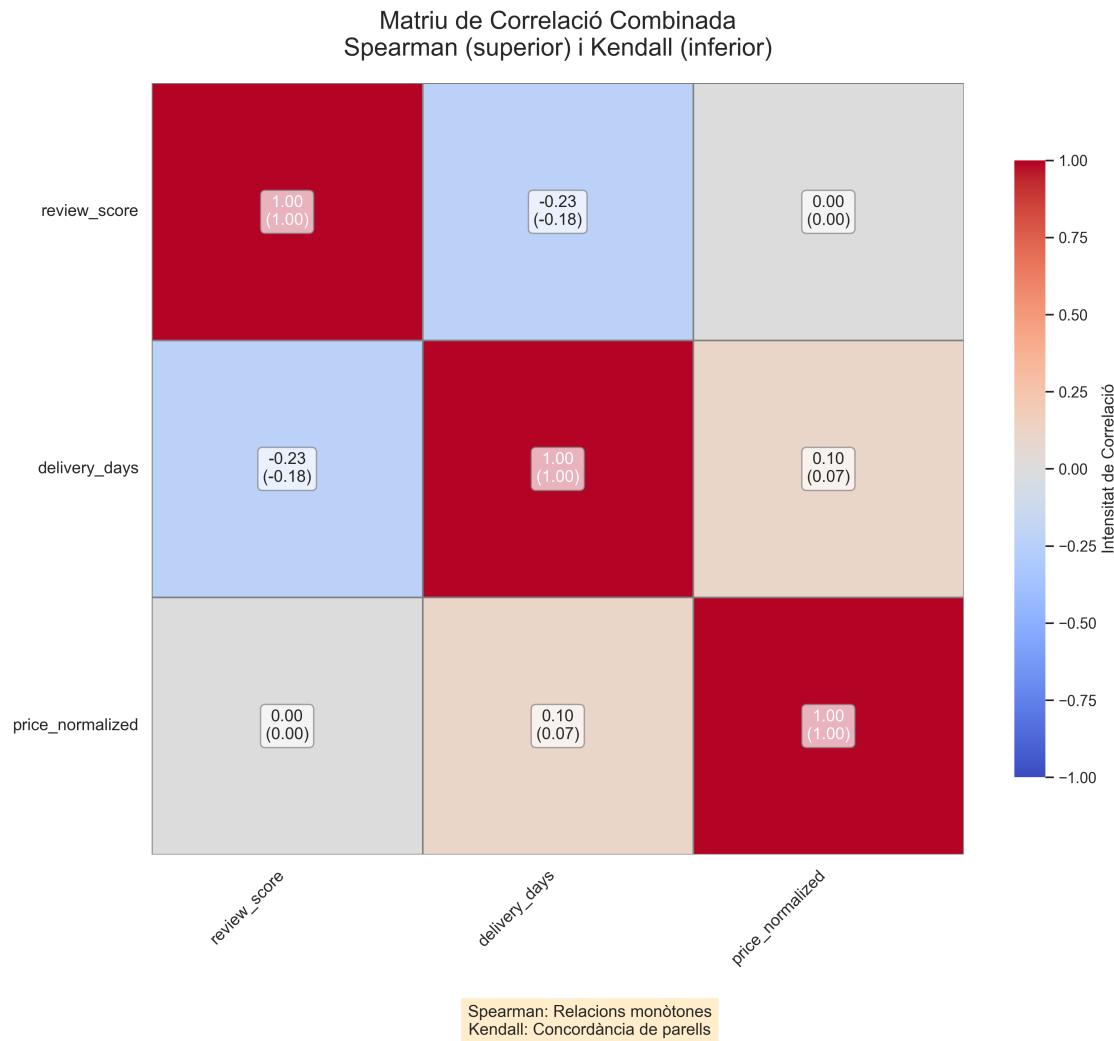
Generant Boxplot de relació de temps lliurament - satisfacció...



Boxplot generat correctament

Generant matriu de correlació millorada...

Matriu de correlació guardada a: G:\Mi unidad\IT ACADEMY\Reskilling Data



CONCLUSIONS DE CORRELACIÓ:

Relació entre Satisfacció del client i Dies de lliurament:

- Spearman: -0.228, Kendall: -0.180
- Spearman: Correlació feble i negativa (quan Satisfacció del client augmenta, Dies de lliurament tendeix a disminuir)
- Kendall: Correlació feble i negativa (quan Satisfacció del client augmenta, Dies de lliurament tendeix a disminuir)

Relació entre Satisfacció del client i Preu normalitzat:

- Spearman: 0.005, Kendall: 0.004
- Spearman: Correlació pràcticament nul·la i positiva (quan Satisfacció del

client augmenta, Preu normalitzat tendeix a augmentar)

- Kendall: Correlació pràcticament nul·la i positiva (quan Satisfacció del client augmenta, Preu normalitzat tendeix a augmentar)

Relació entre Dies de lliurament i Preu normalitzat:

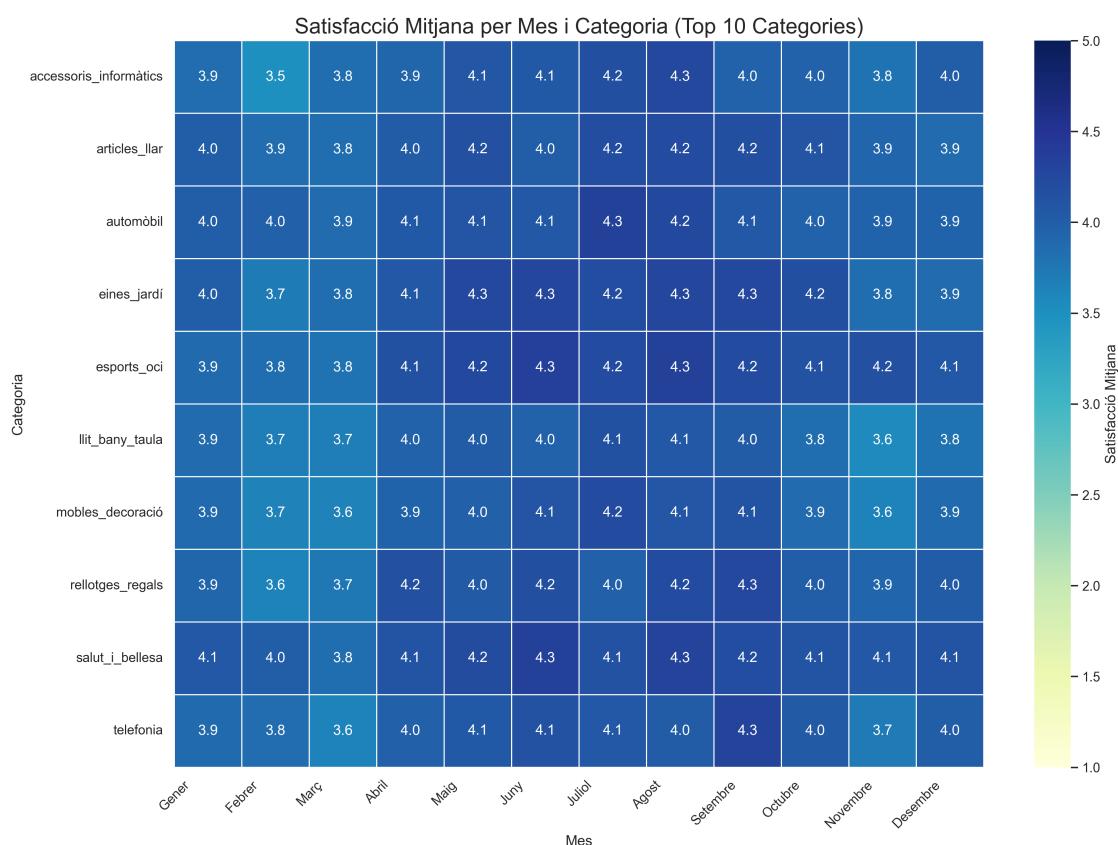
- Spearman: 0.102, Kendall: 0.070

- Spearman: Correlació feble i positiva (quan Dies de lliurament augmenta, Preu normalitzat tendeix a augmentar)

- Kendall: Correlació pràcticament nul·la i positiva (quan Dies de lliurament augmenta, Preu normalitzat tendeix a augmentar)

Matriu de correlació generada correctament

Generant matriu de satisfacció per mes i categoria...



Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\visualitzacions\matriu_satisfaccio_mes_categoria.png

Estadístiques de la matriu:

- Categories incloses: 10

- Mesos inclosos: 12

- Valors calculats: 120

- Valors NaN: 0

Temps d'execució de l'anàlisi exploratòria: 45.35 segons

FASE 3: ANÀLISI ESTRATÈGIC

FASE 3.2. ANÀLISI DE PREU I DEMANDA

ANÀLISI DE PREU I DEMANDA

Preparant dades per a anàlisi de preu i demanda...

S'han filtrat 775 registres amb preu 0 o negatiu

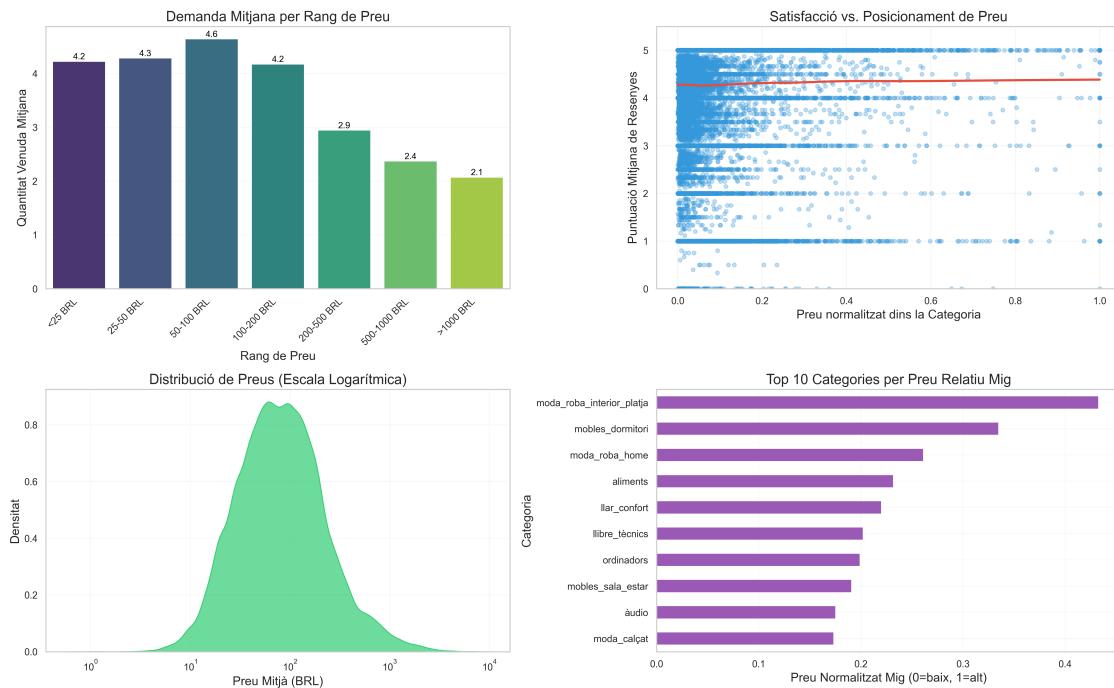
Agregant dades per producte i categoria...

Calculant rangs de preu i percentils...

Generant visualitzacions...

Visualitzacions guardades a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\visualitzacions\analisi_preu_demanda.png

Anàlisi de Preu i Demanda



INSIGHTS CLAU I RECOMANACIONS ESTRATÈGIQUES

PRODUCTES MÉS VENUTS (per unitats):

categoria_catala	preu_mitja	quantitat_total
eines_jardí	54.912	793
mobles_decoració	71.364	640
eines_jardí	54.270	551
eines_jardí	54.657	545
llit_bany_taula	88.167	542

PRODUCTES PREMIUMS AMB ALTA DEMANDA:

categoria_catala	preu_mitja	quantitat_total
esports_oci	3874.500	2
ordinadors	3133.323	3
accessoris_informàtics	2999.890	5
accessoris_informàtics	2649.990	2
productes_interessants	2649.000	2

PRODUCTES D'ALTA QUALITAT I BAIX PREU:

categoria_catala	preu_mitja	mitjana_review	quantitat_total
productes_interessants	39.000	5.000	21
articles_llar	20.194	5.000	80
llit_bany_taula	33.500	5.000	16
productes_interessants	11.990	5.000	12
salut_i_bellesa	21.323	5.000	11

RECOMANACIONS ESTRATÈGIQUES:

1. ESTRATÈGIA PER A PRODUCTES MASSIUS:

- Centrar-se en la categoria 'eines_jardí' que representa el volum més alt de vendes
- Optimitzar l'inventari i la logística per a aquests productes d'alta rotació
- Crear paquets o ofertes combinades per augmentar la mitjana de venda
- Utilitzar com a 'productes ganxo' per atraure nous clients

2. ESTRATÈGIA PER A PRODUCTES PREMIUM:

- Desenvolupar campanyes de màrqueting exclusives per a productes d'alta gamma
- Millorar la presentació i descripcions d'aquests productes amb contingut multimèdia
- Ofereix garanties esteses o serveis postvenda premium
- Segmentar publicitat cap a clients d'alt poder adquisitiu

3. ESTRATÈGIA PER A PRODUCTES D'ALTA QUALITAT-PREU:

- Promocionar agressivament productes com 'productes_interessants' (BRL39.00)
- 5.0/5)
- Destacar aquests productes a les pàgines d'inici i cerques

- Utilitzar com a testimonis de qualitat de la botiga
- Considerar augmentar lleugerament el preu sense superar el 10%

4. ESTRATÈGIES TRANSVERSALS:

- Implementar preus dinàmics basats en la demanda estacional
- Crear programes de fidelització per a compradors freqüents de productes premium
- Fer tests A/B per a diferents estratègies de preu per categoria
- Desenvolupar bundles que combinin productes massius amb d'alt marge

5. OPORTUNITATS PER CATEGORIA:

- Categories amb major marge: ordinadors, agroindústria_comerç, electrodomèstics_petits
- Considerar ampliar l'assortiment en aquestes categories

Temps d'execució de l'anàlisi: 50.89 segons

FASE 3.3. IMPACTE DE FOTOS EN PRODUCTES

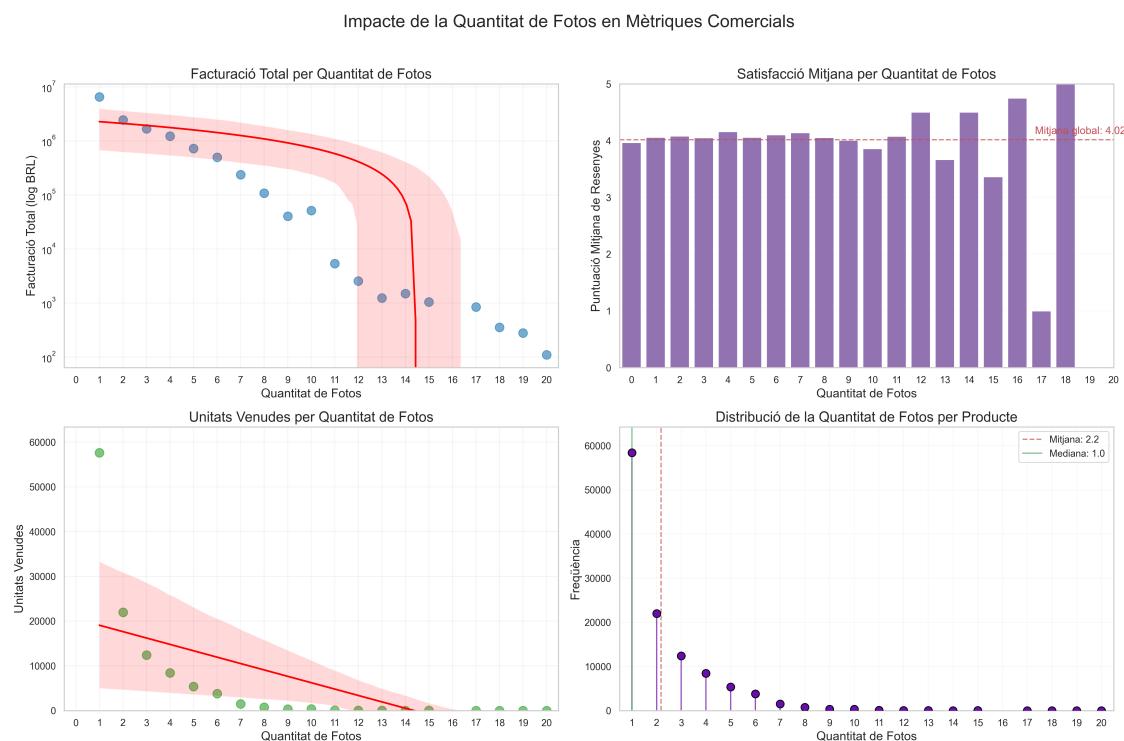
ANÀLISI DE L'IMPACTE DE LES FOTOS DEL PRODUCTE

MÈTRIQUES CLAU PER QUANTITAT DE FOTOS:

	product_photos_qty	num_productes	facturacio_total	mitjana_review
	unitats_venudes			
57646	1.000	17106	6555966.500	3.965
21962	2.000	6262	2449236.000	4.059
12389	3.000	3859	1682596.500	4.078
8430	4.000	2425	1227725.500	4.051
5365	5.000	1483	725333.625	4.157
3785	6.000	967	499250.750	4.058
1501	7.000	343	238081.859	4.101
727	8.000	192	108374.969	4.138
313	9.000	105	40301.871	4.056
342	10.000	95	51516.770	4.007
71	11.000	46	5342.270	3.859

	12.000	35	2551.980	4.075
53	13.000	9	1236.780	4.500
30	14.000	5	1500.870	3.667
6	15.000	8	1043.770	4.500
12	17.000	7	841.000	3.364
11	18.000	2	354.400	4.750
4	19.000	1	277.880	1.000
2	20.000	1	110.270	5.000
1				

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\visualitzacions\impacte_fotos_producte.png



ANÀLISI DE CORRELACIÓ:

	product_photos_qty	price	review_score	order_item_id
product_photos_qty	1.000	0.051	0.031	-0.055

price	0.051	1.000	-0.004	-0.061
review_score	0.031	-0.004	1.000	-0.130
order_item_id	-0.055	-0.061	-0.130	1.000

RECOMANACIONS ESTRATÈGIQUES:

1. Productes amb 3-6 fotos tendeixen a tenir millors resultats comercials
 2. Un excés de fotos (>10) no millora la satisfacció del client
 3. Optimitzi la qualitat de les fotos en comptes de la quantitat
 4. Realitzi tests A/B per determinar el nombre òptim de fotos per categoria
- Temps d'execució de l'anàlisi: 3.87 segons

Temps d'execució de l'anàlisi estratègica: 54.75 segons

FASE 4: MODELATGE PREDICTIU

FASE 4.1. PREPARACIÓ DE DADES PER MODELATGE

Preparant dades per a model predictiu...

Dades preparades amb 101 setmanes i 52 categories

FASE 4.2. ENTENAMENT DE MODELS PER CATEGORIA

ENTENANT MODELS PER A 15 CATEGORIES PRINCIPALS

Entrenant models per categoria: 0% | 0/15 [00:00<?, ?it/s] 11:59:22 - cmdstanpy - INFO - Chain [1] start processing

Entrenant model Prophet per a llit_bany_taula...

11:59:23 - cmdstanpy - INFO - Chain [1] done processing

Model entrenat per a llit_bany_taula en 0.58s

- MAE: 20.26, MAPE: 62.28%, R²: 0.7957
- MSE: 793.53, RMSE: 28.17, MedAE: 15.42, MaxError: 131.53

Entrenant models per categoria: 7% | 1/15 [00:01<00:17, 1.27s/it]

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public

Dataset\prediccions\categories\prediccions_llit_bany_taula.png

Entrenant XGBoost per a llit_bany_taula...

XGBoost entrenat per a llit_bany_taula

- MAE: 74.33, MAPE: 104.10%, R²: -1.0298

Entrenant SARIMA per a llit_bany_taula...

SARIMA entrenat per a llit_bany_taula

- MAE: 21.52, MAPE: 29.55%, R²: 0.6157

Entrenant model Prophet per a salut_i_bellesa...

```
11:59:24 - cmdstanpy - INFO - Chain [1] start processing
11:59:24 - cmdstanpy - INFO - Chain [1] done processing
```

Model entrenat per a salut_i_bellesa en 0.30s
- MAE: 16.46, MAPE: 44.21%, R²: 0.8693
- MSE: 538.19, RMSE: 23.20, MedAE: 11.10, MaxError: 85.84

```
Entrenant models per categoria: 13% | 2/15 [00:02<00:14, 1.12s/it]
```

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT
10\Brazilian E-Commerce Public
Dataset\prediccions\categories\prediccions_salut_i_bellesa.png

Entrenant XGBoost per a salut_i_bellesa...
XGBoost entrenat per a salut_i_bellesa
- MAE: 82.20, MAPE: 90.70%, R²: -0.3434
Entrenant SARIMA per a salut_i_bellesa...
SARIMA entrenat per a salut_i_bellesa
- MAE: 19.52, MAPE: 36.91%, R²: 0.7625
Entrenant model Prophet per a esports_oci...

```
11:59:25 - cmdstanpy - INFO - Chain [1] start processing
11:59:25 - cmdstanpy - INFO - Chain [1] done processing
```

Model entrenat per a esports_oci en 0.25s
- MAE: 20.41, MAPE: 49.73%, R²: 0.7401
- MSE: 647.51, RMSE: 25.45, MedAE: 18.26, MaxError: 72.86

```
Entrenant models per categoria: 20% | 3/15 [00:03<00:12, 1.04s/it]
```

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT
10\Brazilian E-Commerce Public
Dataset\prediccions\categories\prediccions_esports_oci.png

Entrenant XGBoost per a esports_oci...
XGBoost entrenat per a esports_oci
- MAE: 65.59, MAPE: 96.35%, R²: -1.9178
Entrenant SARIMA per a esports_oci...
SARIMA entrenat per a esports_oci
- MAE: 15.48, MAPE: 31.90%, R²: 0.7838
Entrenant model Prophet per a accessoris_informàtics...

```
11:59:26 - cmdstanpy - INFO - Chain [1] start processing
11:59:26 - cmdstanpy - INFO - Chain [1] done processing
```

Model entrenat per a accessoris_informàtics en 0.32s
- MAE: 13.54, MAPE: 31.86%, R²: 0.8985
- MSE: 283.62, RMSE: 16.84, MedAE: 11.54, MaxError: 46.09

```
Entrenant models per categoria: 27% | 4/15 [00:04<00:11, 1.04s/it]
```

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT
10\Brazilian E-Commerce Public
Dataset\prediccions\categories\prediccions_accessoris_informàtics.png

Entrenant XGBoost per a accessoris_informàtics...

XGBoost entrenat per a accessoris_informàtics
 - MAE: 49.08, MAPE: 140.72%, R²: -0.9554
 Entrenant SARIMA per a accessoris_informàtics...
 SARIMA entrenat per a accessoris_informàtics
 - MAE: 16.11, MAPE: 40.81%, R²: 0.7772
 Entrenant model Prophet per a mobles_decoració...

11:59:27 - cmdstanpy - INFO - Chain [1] start processing
 11:59:27 - cmdstanpy - INFO - Chain [1] done processing

Model entrenat per a mobles_decoració en 0.23s
 - MAE: 14.33, MAPE: 76.41%, R²: 0.7646
 - MSE: 368.12, RMSE: 19.19, MedAE: 10.95, MaxError: 62.03

Entrenant models per categoria: 33% | 5/15 [00:05<00:10, 1.00s/it]

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\prediccions\categories\prediccions_mobles_decoració.png

Entrenant XGBoost per a mobles_decoració...
 XGBoost entrenat per a mobles_decoració
 - MAE: 75.25, MAPE: 498.12%, R²: -3.3179
 Entrenant SARIMA per a mobles_decoració...
 SARIMA entrenat per a mobles_decoració
 - MAE: 16.45, MAPE: 56.08%, R²: 0.5779
 Entrenant model Prophet per a articles_llar...

11:59:28 - cmdstanpy - INFO - Chain [1] start processing
 11:59:28 - cmdstanpy - INFO - Chain [1] done processing

Model entrenat per a articles_llar en 0.33s
 - MAE: 11.44, MAPE: 44.93%, R²: 0.8561
 - MSE: 253.59, RMSE: 15.92, MedAE: 8.46, MaxError: 59.04

Entrenant models per categoria: 40% | 6/15 [00:06<00:09, 1.01s/it]

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\prediccions\categories\prediccions_articles_llar.png

Entrenant XGBoost per a articles_llar...
 XGBoost entrenat per a articles_llar
 - MAE: 61.58, MAPE: 75.00%, R²: -0.5320
 Entrenant SARIMA per a articles_llar...
 SARIMA entrenat per a articles_llar
 - MAE: 14.70, MAPE: 39.52%, R²: 0.7231
 Entrenant model Prophet per a rellotges_regals...

11:59:29 - cmdstanpy - INFO - Chain [1] start processing
 11:59:29 - cmdstanpy - INFO - Chain [1] done processing

Model entrenat per a rellotges_regals en 0.27s
 - MAE: 11.69, MAPE: 36.86%, R²: 0.8890
 - MSE: 249.92, RMSE: 15.81, MedAE: 8.83, MaxError: 46.05

Entrenant models per categoria: 47% | 7/15 [00:07<00:08, 1.00s/it]

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public

Dataset\prediccions\categories\prediccions_rellotges_regals.png

Entrenant XGBoost per a rellotges_regals...

XGBoost entrenat per a rellotges_regals

- MAE: 54.00, MAPE: 179.14%, R²: -0.8691

Entrenant SARIMA per a rellotges_regals...

SARIMA entrenat per a rellotges_regals

- MAE: 14.62, MAPE: 40.34%, R²: 0.7677

Entrenant model Prophet per a telefonia...

11:59:30 - cmdstanpy - INFO - Chain [1] start processing

11:59:30 - cmdstanpy - INFO - Chain [1] done processing

Model entrenat per a telefonia en 0.28s

- MAE: 11.66, MAPE: 71.24%, R²: 0.7409

- MSE: 206.18, RMSE: 14.36, MedAE: 11.13, MaxError: 40.96

Entrenant models per categoria: 53% | 8/15 [00:08<00:07, 1.03s/it]

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public

Dataset\prediccions\categories\prediccions_telefonia.png

Entrenant XGBoost per a telefonia...

XGBoost entrenat per a telefonia

- MAE: 31.58, MAPE: 158.90%, R²: -0.8158

Entrenant SARIMA per a telefonia...

SARIMA entrenat per a telefonia

- MAE: 10.68, MAPE: 39.12%, R²: 0.6602

Entrenant model Prophet per a automòbil...

11:59:31 - cmdstanpy - INFO - Chain [1] start processing

11:59:31 - cmdstanpy - INFO - Chain [1] done processing

Model entrenat per a automòbil en 0.26s

- MAE: 7.38, MAPE: 25.75%, R²: 0.9054

- MSE: 89.77, RMSE: 9.47, MedAE: 5.75, MaxError: 23.70

Entrenant models per categoria: 60% | 9/15 [00:09<00:06, 1.02s/it]

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public

Dataset\prediccions\categories\prediccions_automòbil.png

Entrenant XGBoost per a automòbil...

XGBoost entrenat per a automòbil

- MAE: 47.03, MAPE: 136.26%, R²: -1.1026

Entrenant SARIMA per a automòbil...

SARIMA entrenat per a automòbil

- MAE: 10.22, MAPE: 44.10%, R²: 0.7198

Entrenant model Prophet per a joguines...

```
11:59:32 - cmdstanpy - INFO - Chain [1] start processing
11:59:32 - cmdstanpy - INFO - Chain [1] done processing
```

Model entrenat per a joguines en 0.29s
- MAE: 7.55, MAPE: 34.63%, R²: 0.9118
- MSE: 94.95, RMSE: 9.74, MedAE: 5.33, MaxError: 34.63

```
Entrenant models per categoria: 67% | 10/15 [00:10<00:05, 1.02s/it]
```

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public
Dataset\prediccions\categories\prediccions_joguines.png

Entrenant XGBoost per a joguines...
XGBoost entrenat per a joguines
- MAE: 33.68, MAPE: 100.63%, R²: -4.8978
Entrenant SARIMA per a joguines...
SARIMA entrenat per a joguines
- MAE: 10.15, MAPE: 40.22%, R²: 0.7090
Entrenant model Prophet per a productes_interessants...

```
11:59:33 - cmdstanpy - INFO - Chain [1] start processing
11:59:33 - cmdstanpy - INFO - Chain [1] done processing
```

Model entrenat per a productes_interessants en 0.27s
- MAE: 14.00, MAPE: 99.85%, R²: 0.4542
- MSE: 275.34, RMSE: 16.59, MedAE: 13.39, MaxError: 37.31

```
Entrenant models per categoria: 73% | 11/15 [00:11<00:04, 1.02s/it]
```

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public

Dataset\prediccions\categories\prediccions_productes_interessants.png

Entrenant XGBoost per a productes_interessants...
XGBoost entrenat per a productes_interessants
- MAE: 31.54, MAPE: 190.66%, R²: -5.4112
Entrenant SARIMA per a productes_interessants...
SARIMA entrenat per a productes_interessants
- MAE: 8.20, MAPE: 30.22%, R²: 0.7502
Entrenant model Prophet per a eines_jardí...

```
11:59:34 - cmdstanpy - INFO - Chain [1] start processing
11:59:34 - cmdstanpy - INFO - Chain [1] done processing
```

Model entrenat per a eines_jardí en 0.38s
- MAE: 9.45, MAPE: 52.32%, R²: 0.7218
- MSE: 219.19, RMSE: 14.81, MedAE: 6.78, MaxError: 89.21

```
Entrenant models per categoria: 80% | 12/15 [00:12<00:03, 1.04s/it]
```

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public

Dataset\prediccions\categories\prediccions_eines_jardi.png

Entrenant XGBoost per a eines_jardí...

XGBoost entrenat per a eunes_jardí
 - MAE: 41.53, MAPE: 542.78%, R²: -5.8611
 Entrenant SARIMA per a eunes_jardí...
 SARIMA entrenat per a eunes_jardí
 - MAE: 13.27, MAPE: 49.98%, R²: 0.2558
 Entrenant model Prophet per a perfumeria...

11:59:35 - cmdstanpy - INFO - Chain [1] start processing
 11:59:35 - cmdstanpy - INFO - Chain [1] done processing

Model entrenat per a perfumeria en 0.89s
 - MAE: 8.91, MAPE: 88.44%, R²: 0.7330
 - MSE: 120.64, RMSE: 10.98, MedAE: 7.84, MaxError: 33.79

Entrenant models per categoria: 87% | 13/15 [00:14<00:02, 1.22s/it]

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\prediccions\categories\prediccions_perfumeria.png

Entrenant XGBoost per a perfumeria...
 XGBoost entrenat per a perfumeria
 - MAE: 23.88, MAPE: 311.62%, R²: -0.6781
 Entrenant SARIMA per a perfumeria...
 SARIMA entrenat per a perfumeria
 - MAE: 8.92, MAPE: 112.00%, R²: 0.5792
 Entrenant model Prophet per a nadons...

11:59:36 - cmdstanpy - INFO - Chain [1] start processing
 11:59:37 - cmdstanpy - INFO - Chain [1] done processing

Model entrenat per a nadons en 0.32s
 - MAE: 7.17, MAPE: 59.54%, R²: 0.7801
 - MSE: 90.90, RMSE: 9.53, MedAE: 5.57, MaxError: 34.35

Entrenant models per categoria: 93% | 14/15 [00:15<00:01, 1.18s/it]

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\prediccions\categories\prediccions_nadons.png

Entrenant XGBoost per a nadons...
 XGBoost entrenat per a nadons
 - MAE: 29.62, MAPE: 155.76%, R²: -1.1358
 Entrenant SARIMA per a nadons...
 SARIMA entrenat per a nadons
 - MAE: 8.26, MAPE: 40.83%, R²: 0.6828
 Entrenant model Prophet per a altres...

11:59:38 - cmdstanpy - INFO - Chain [1] start processing
 11:59:38 - cmdstanpy - INFO - Chain [1] done processing

Model entrenat per a altres en 0.28s
 - MAE: 9.08, MAPE: 119.86%, R²: 0.5860
 - MSE: 135.62, RMSE: 11.65, MedAE: 8.06, MaxError: 38.14

Entrenant models per categoria: 100% | 15/15 [00:16<00:00, 1.08s/it]

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\prediccions\categories\prediccions_altres.png

Entrenant XGBoost per a altres...

XGBoost entrenat per a altres

- MAE: 17.42, MAPE: 373.07%, R²: -1.0120

Entrenant SARIMA per a altres...

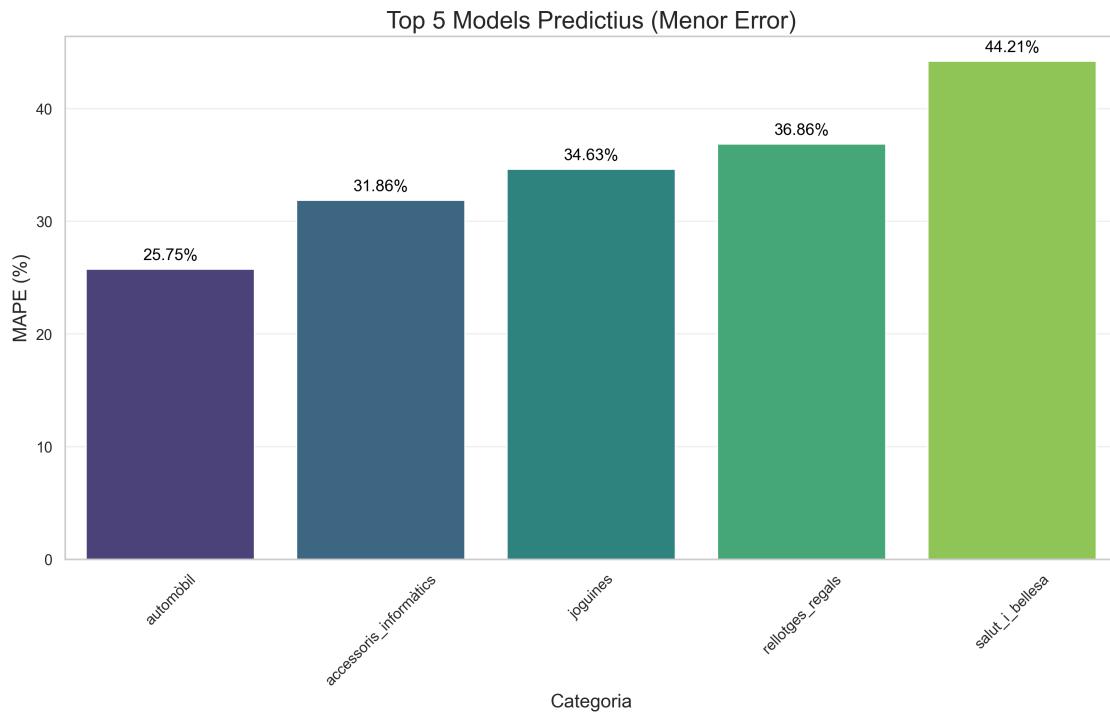
SARIMA entrenat per a altres

- MAE: 9.24, MAPE: 45.16%, R²: 0.5585

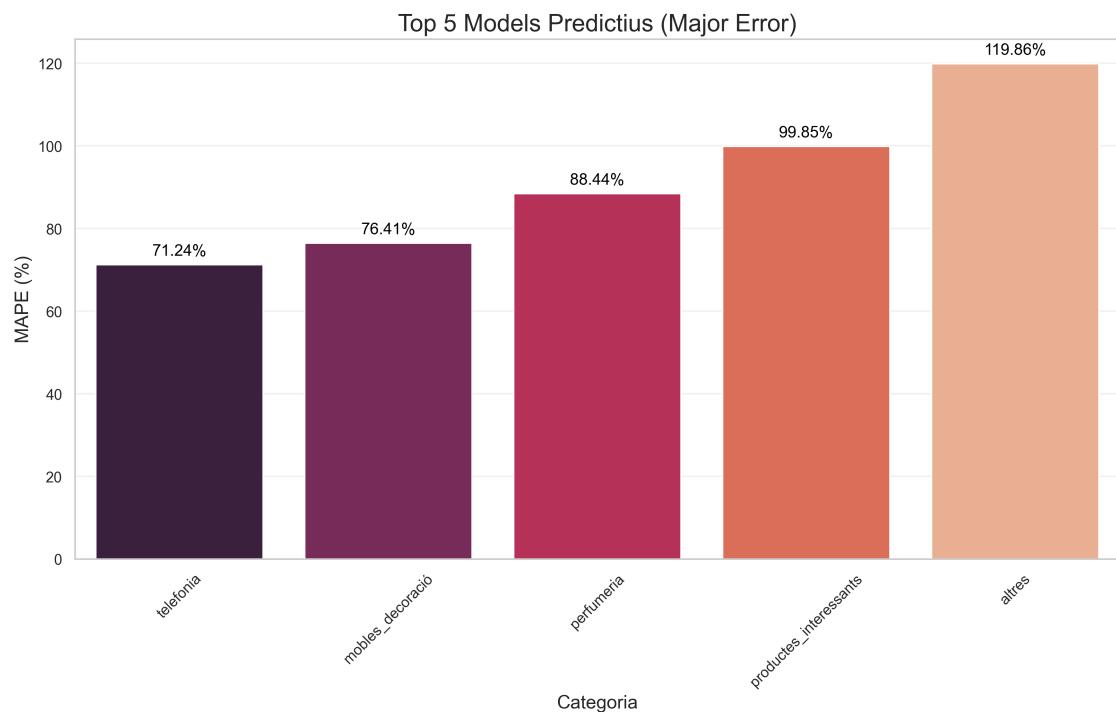
FASE 4.3. AVALUACIÓ I COMPARATIVA DE MODELS

RESULTATS DELS MODELS

Gràfic dels 5 millors models guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\prediccions\top5_millors_models.png

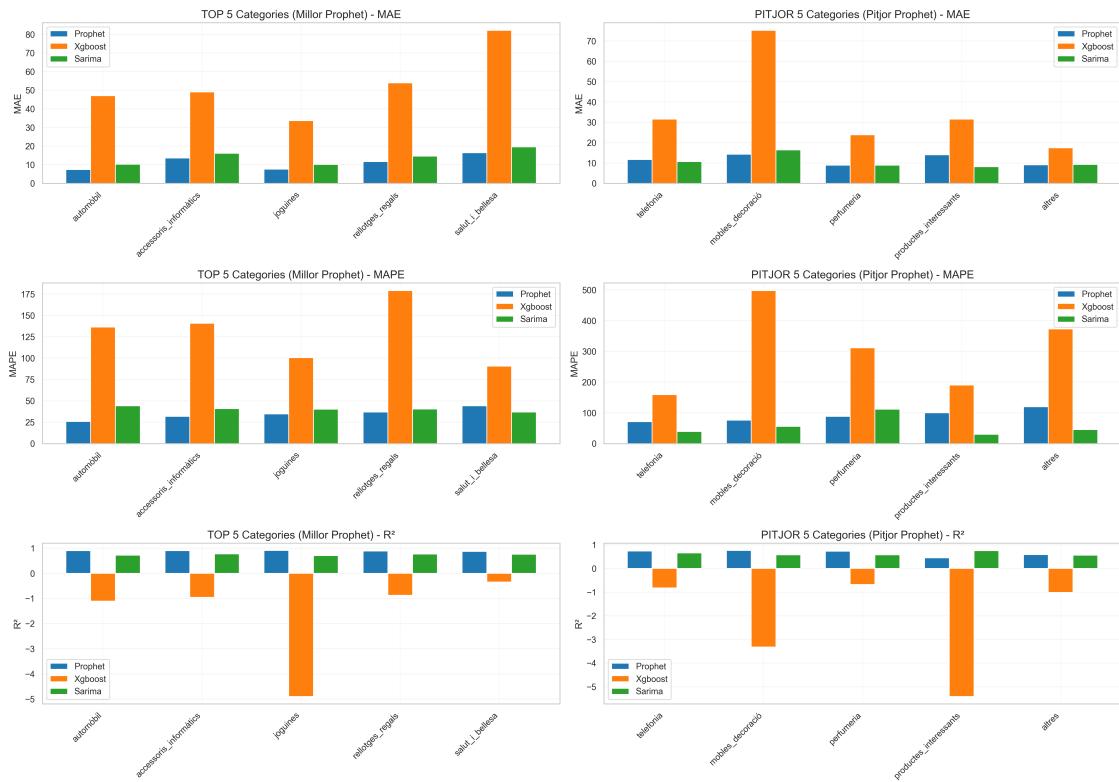


Gràfic dels 5 pitjors models guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\prediccions\top5_pitjors_models.png



Comparativa detallada guardada a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\prediccions\comparativa_top_pitjor_detail.png

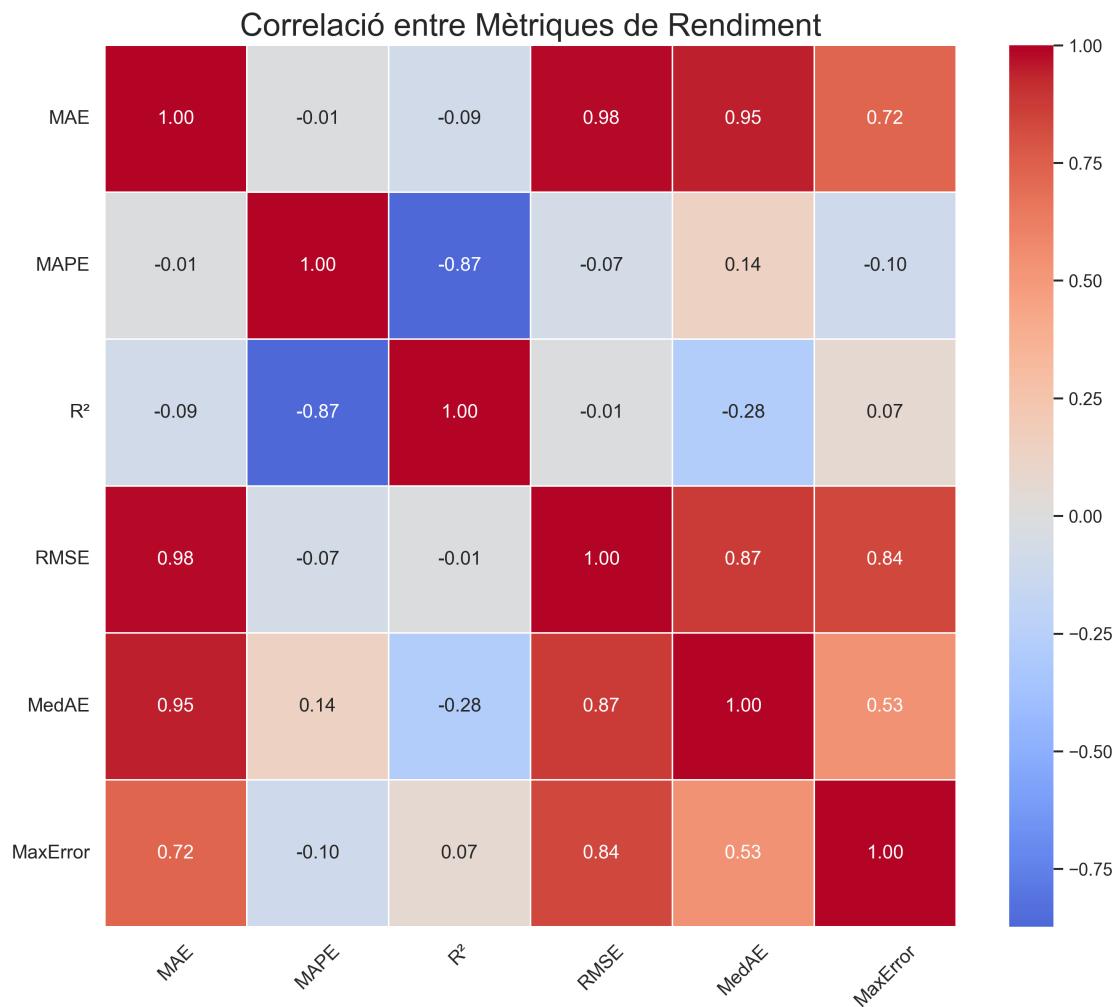
Comparativa Detallada de Models per Categories Destacades



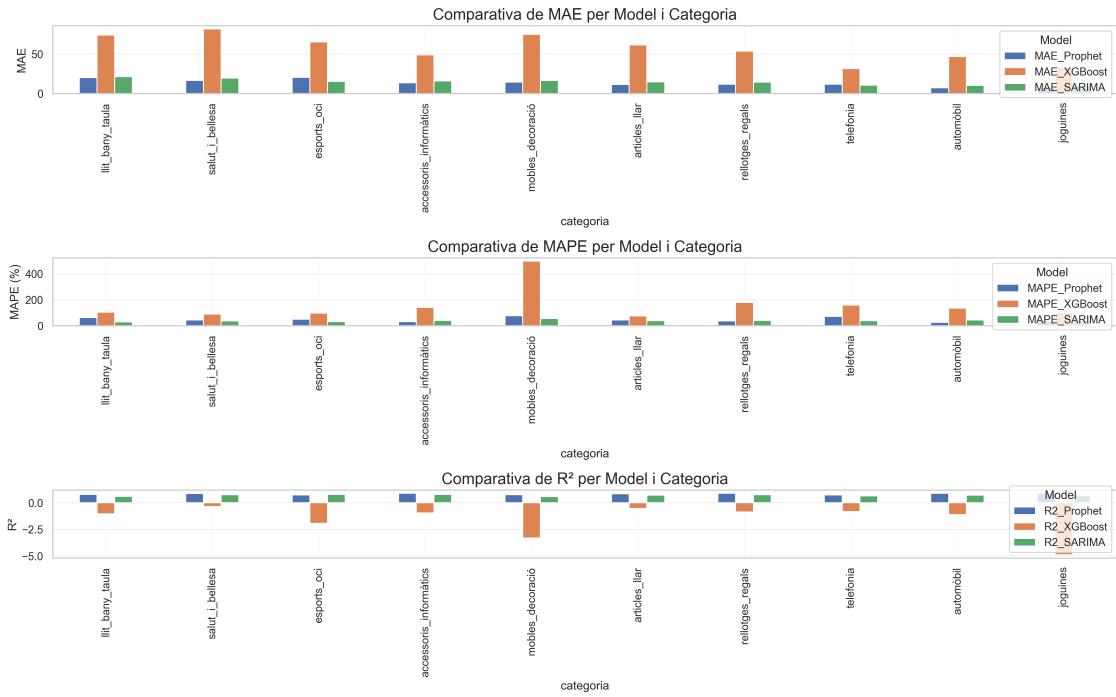
Mètriques de Rendiment per a les 15 Categories Principals



Gràfic de mètriques guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\prediccions\metrics_top_15_categories.png



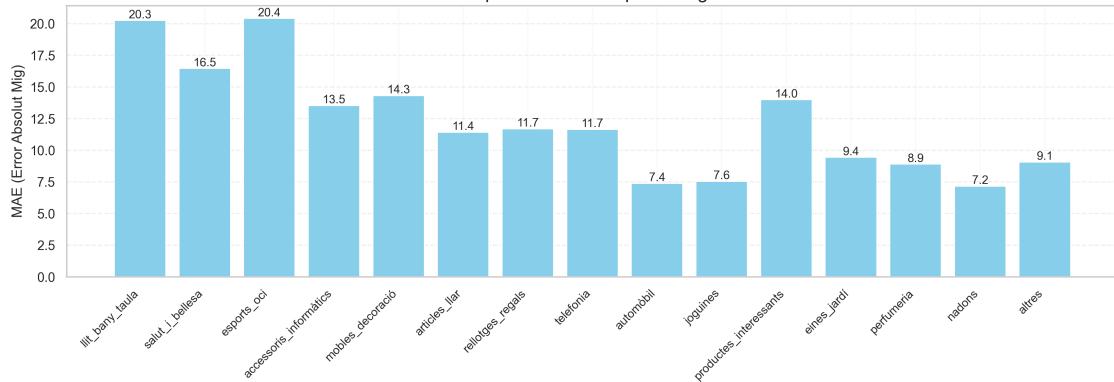
Matriu de correlació guardada a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\prediccions\correlacio_metrics.png



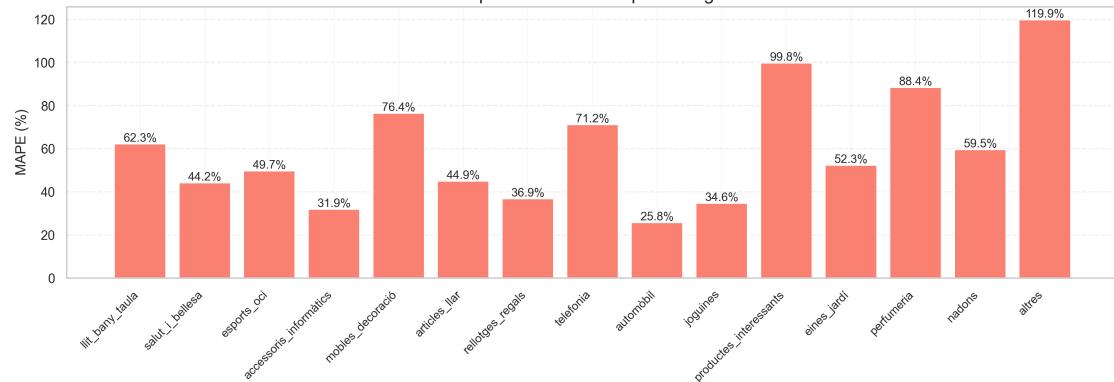
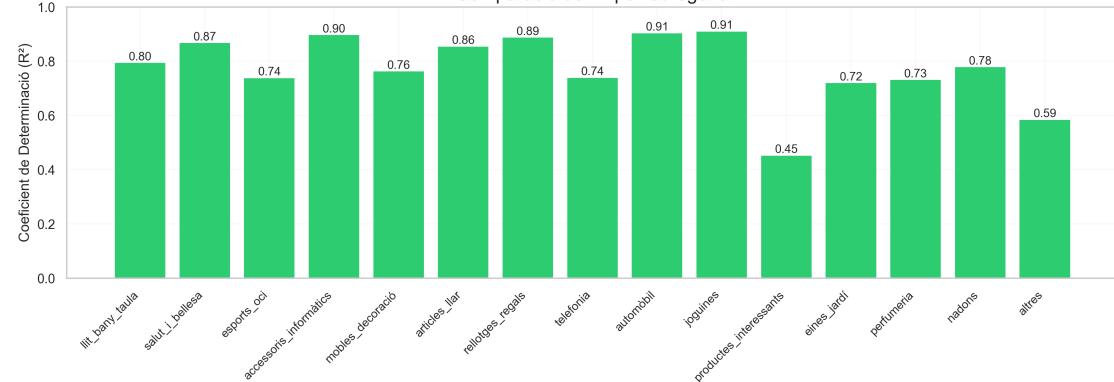
Comparativa de models guardada a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\prediccions\comparacio_models_categories.png

Gràfic guardat a: G:\Mi unidad\IT ACADEMY\Reskilling Data Analytics\SPRINT 10\Brazilian E-Commerce Public Dataset\prediccions\resum_comparatiu_categories.png

Comparació de MAE per Categoria



Comparació de MAPE per Categoria

Comparació de R² per Categoria

RESUM COMPARATIU GUARDAT I MOSTRAT

RENDIMENT PER CATEGORIA:

- llit_bany_taula: MAE = 20.26, MAPE = 62.28%, R² = 0.7957
- salut_i_belleza: MAE = 16.46, MAPE = 44.21%, R² = 0.8693
- esports_ocl: MAE = 20.41, MAPE = 49.73%, R² = 0.7401
- accessoris_informàticas: MAE = 13.54, MAPE = 31.86%, R² = 0.8985
- mobles_decoració: MAE = 14.33, MAPE = 76.41%, R² = 0.7646

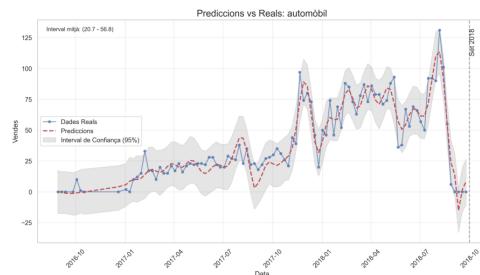
- articles_llar: MAE = 11.44, MAPE = 44.93%, R² = 0.8561
 - rellotges_regals: MAE = 11.69, MAPE = 36.86%, R² = 0.8890
 - telefonia: MAE = 11.66, MAPE = 71.24%, R² = 0.7409
 - automòbil: MAE = 7.38, MAPE = 25.75%, R² = 0.9054
 - joguines: MAE = 7.55, MAPE = 34.63%, R² = 0.9118
 - productes_interessants: MAE = 14.00, MAPE = 99.85%, R² = 0.4542
 - eines_jardí: MAE = 9.45, MAPE = 52.32%, R² = 0.7218
 - perfumeria: MAE = 8.91, MAPE = 88.44%, R² = 0.7330
 - nadons: MAE = 7.17, MAPE = 59.54%, R² = 0.7801
 - altres: MAE = 9.08, MAPE = 119.86%, R² = 0.5860
-

FASE 4.4. VISUALITZACIÓ DE RESULTATS

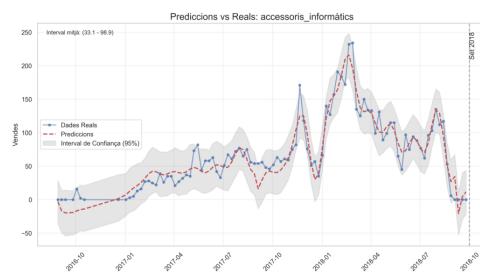
VISUALITZANT PREDICCIIONS

Predictions dels 5 Millors Models

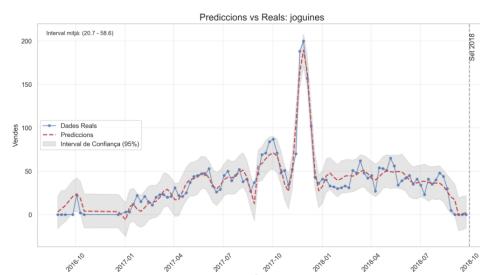
automòbil (MAPE: 25.75%)



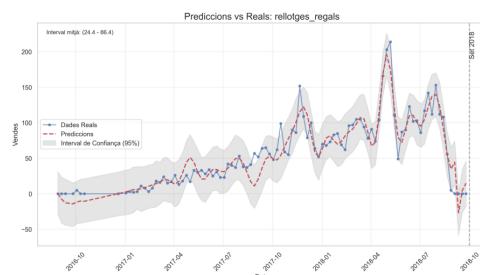
accessoris_informàtics (MAPE: 31.86%)



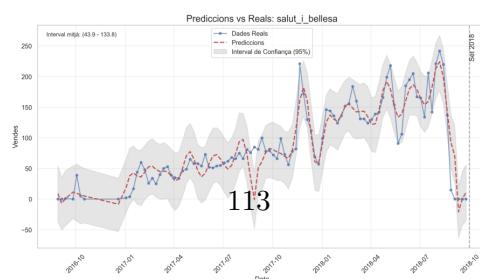
joguines (MAPE: 34.63%)



rellotges_regals (MAPE: 36.86%)

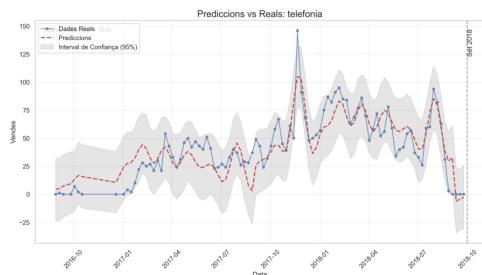


salut_i_belleza (MAPE: 44.21%)

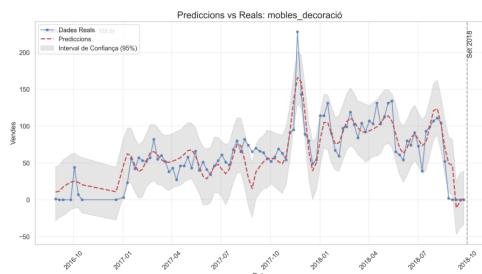


Prediccions dels 5 Pitjors Models

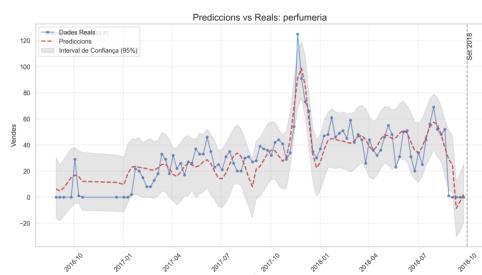
telefonia (MAPE: 71.24%)



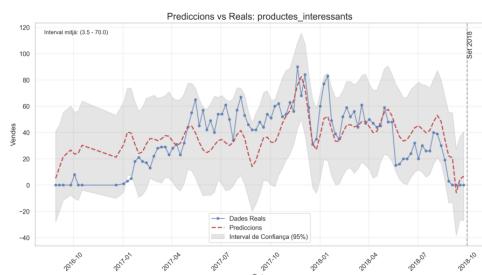
mobles_decoració (MAPE: 76.41%)



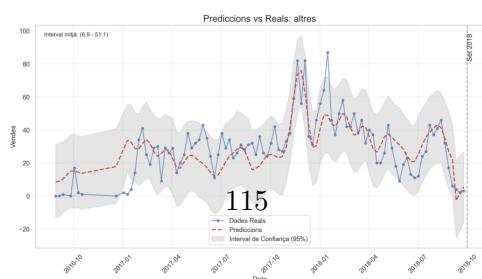
perfumeria (MAPE: 88.44%)



products_interessants (MAPE: 99.85%)



altres (MAPE: 119.86%)



```
Temps d'execució del modelatge predictiu: 41.50 segons
```

```
=====
=====VERIFICACIÓ FINAL DE RESULTATS
```

```
=====VERIFICACIÓ DE VISUALITZACIONS GENERADES
```

```
ESTAT DE VISUALITZACIONS:
```

- comparativa_facturacio_periodes: GENERADA
- comparativa_unitats_periodes: GENERADA
- heatmap_estacionalitat_categories: GENERADA
- comparacio_metrics_models: GENERADA
- correlacio_metrics: GENERADA

```
=====TOTES LES ETAPES COMPLETADES. Temps total: 145.77 segons
```

```
[4]: print("Relació orders-order_items a DF_FINAL:")
print(f"  Ordres úniques: {df_final['order_id'].nunique()}")
print(f"  Registres d'items: {len(df_final)}")
print(f"  Mitjana items/comanda: {len(df_final) / df_final['order_id'].nunique():.2f}")
```

```
Relació orders-order_items a DF_FINAL:
```

```
  Ordres úniques: 99441
```

```
  Registres d'items: 113425
```

```
  Mitjana items/comanda: 1.14
```