vídeos de jogos de videogame

Este relatório apresenta uma análise visual e exploratória dos dados de vendas de jogos eletrônicos entre os anos de 1980 e 2016. O objetivo é compreender os padrões de vendas, a popularidade dos consoles, a frequência anual de lançamentos e o desempenho dos jogos por gênero. A

análise é baseada em gráficos gerados com Python, utilizando bibliotecas como Seaborn, Matplotlib e Plotly.

Para iniciar o projeto, vamos importar as bibliotecas necessárias que utilizaremos:

```
#importando as bibliotecas
import pandas as pd, seaborn as sns, matplotlib.pyplot as plt, numpy as np
import plotly.express as px
import plotly.io as pio
sns.set_palette("deep")
sns.set_style("darkgrid")
```

Vamos agora importar os dados da base de dados .csv:

```
#carregando o dataset
df = pd.read_csv('./dados/vgsales(in).csv', sep=';')
```

O próximo passo foi traduzir as colunas que estão em Inglês:

```
# Traduzindo as colunas para PT-BR
df.columns = ['posicao de venda', 'jogo', 'console', 'ano de lancamento',
'genero', 'publicadora', 'vendas américa do norte', 'vendas europa', 'vendas
japao', 'vendas outras regioes', 'vendas totais']
```

Temos dados que informam ano de lançamento, e caso esses campos não tiverem informações, vamos remover para evitar inconsistências futuras:

```
# Removendo dados sem que não tenham o ano de lançamento informado
df.dropna(subset=['ano de lancamento'], inplace = True)
```

Publicadoras de jogos que não estão identificadas(desconhecidas) estaremos substituindo por ‘desconhecido’:

```
# Registrando como 'Desconhecida' publicadoras faltantes no dataset
df['publicadora'] = df['publicadora'].fillna('desconhecida')
```

Transformando a coluna de dados que informa o ano de lançamento em números inteiros ‘INT’:

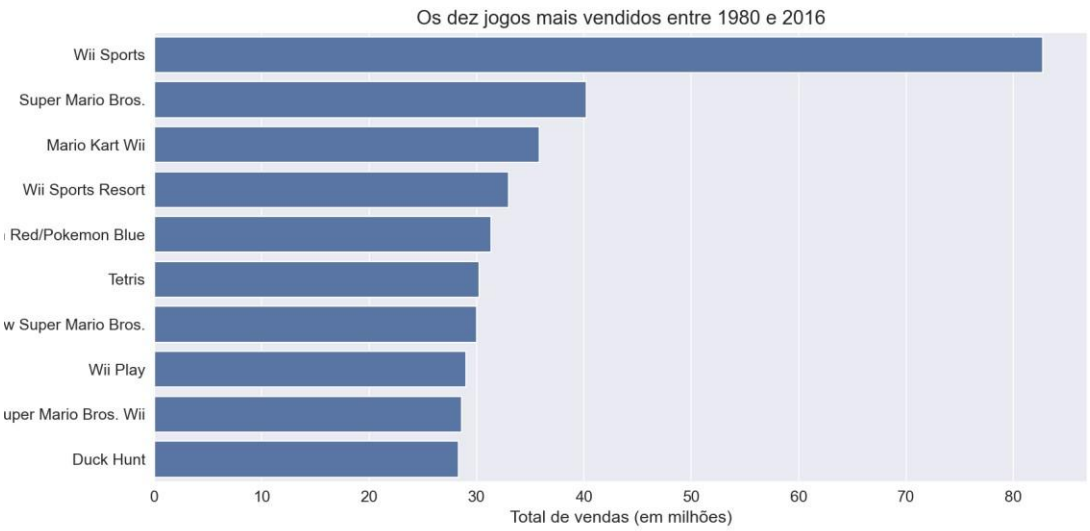
```
# transformando Ano de Lançamento em Int
df['ano de lancamento'] = df['ano de lancamento'].astype(int)
```

1. Jogos Mais Vendidos

O primeiro gráfico apresenta os dez jogos mais vendidos, ordenados pelo total de vendas em milhões. O gráfico de barras horizontal facilita a visualização e comparação entre os títulos.

Identificamos quais jogos tiveram maior impacto no mercado, ajudando a entender as preferências do público.

Jogos icônicos ou de franquias populares normalmente aparecem no topo da lista, refletindo o sucesso comercial e a aceitação.



Registrado abaixo o código fonte utilizado para gerar o gráfico acima:

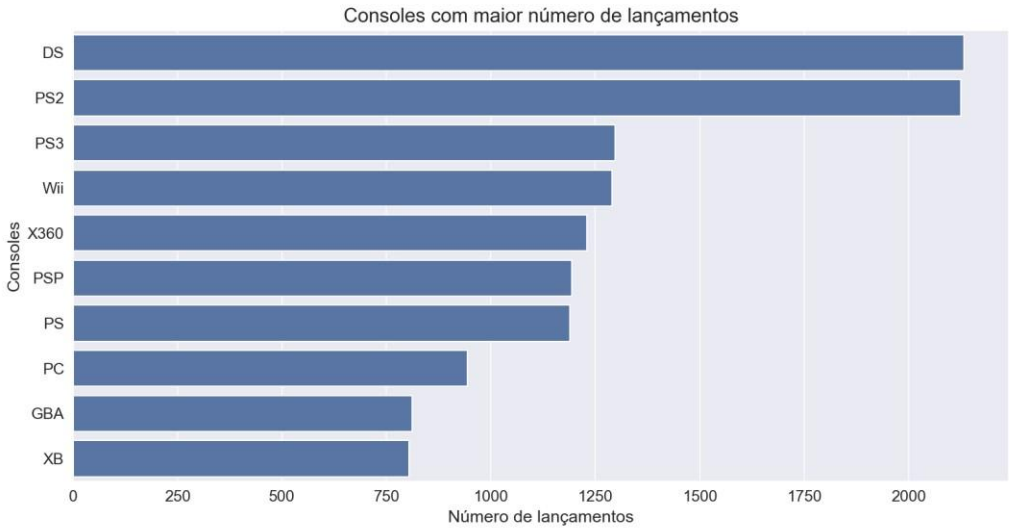
```
top_10 = df.sort_values('vendas totais', ascending=False).head(10)
plt.figure(figsize=(10,7))
sns.barplot(x='vendas totais', y='jogo', data=top_10)
plt.ylabel('Jogos', size=13)
plt.xlabel('Total de vendas (em milhões)', size=13)
plt.title('Os dez jogos mais vendidos entre 1980 e 2016', size=15)
plt.yticks(fontsize=12)
plt.xticks(fontsize=12)
plt.show()
```

2. Consoles com Mais Lançamentos

Este gráfico de barras mostra os consoles com maior número de lançamentos. A contagem dos jogos por console permite avaliar o suporte de desenvolvedores e o volume de oferta para os consumidores.

Consoles com maior número de lançamentos indicam uma base sólida de jogos, o que pode influenciar a popularidade da plataforma.

Observa-se uma concentração dos lançamentos em poucos consoles líderes, refletindo a competição do mercado.



Registrado abaixo o código fonte utilizado para gerar o gráfico acima:

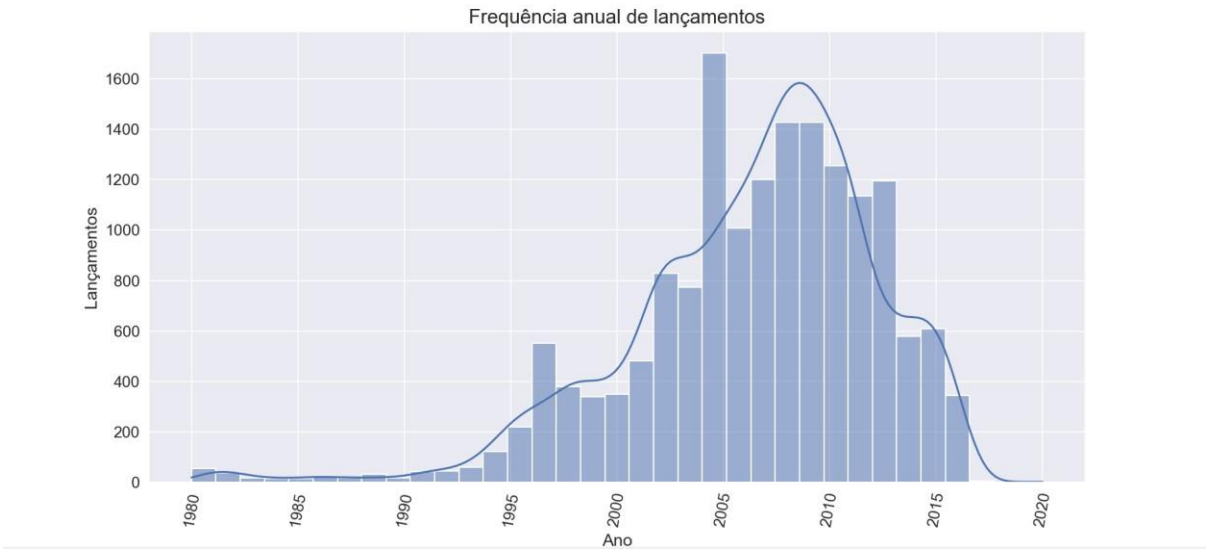
```
most_releases = df['console'].value_counts().head(10).reset_index()
most_releases.columns = ['console', 'lançamentos']
plt.figure(figsize=(10,7))
sns.barplot(x='lançamentos', y='console', data=most_releases)
plt.xlabel('Número de lançamentos', size=13)
plt.ylabel('Consoles', size=13)
plt.title('Consoles com maior número de lançamentos', size=15)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```

3. Frequência Anual de Lançamentos

O histograma com linha KDE mostra a distribuição dos lançamentos de jogos ao longo dos anos. Isso permite identificar picos e períodos de maior ou menor atividade no mercado.

Picos de lançamentos podem estar relacionados a novas gerações de consoles ou eventos especiais na indústria.

Tendências de crescimento ou declínio nas ofertas anuais ajudam a entender o dinamismo do mercado.



Registrado abaixo o código fonte utilizado para gerar o gráfico acima:

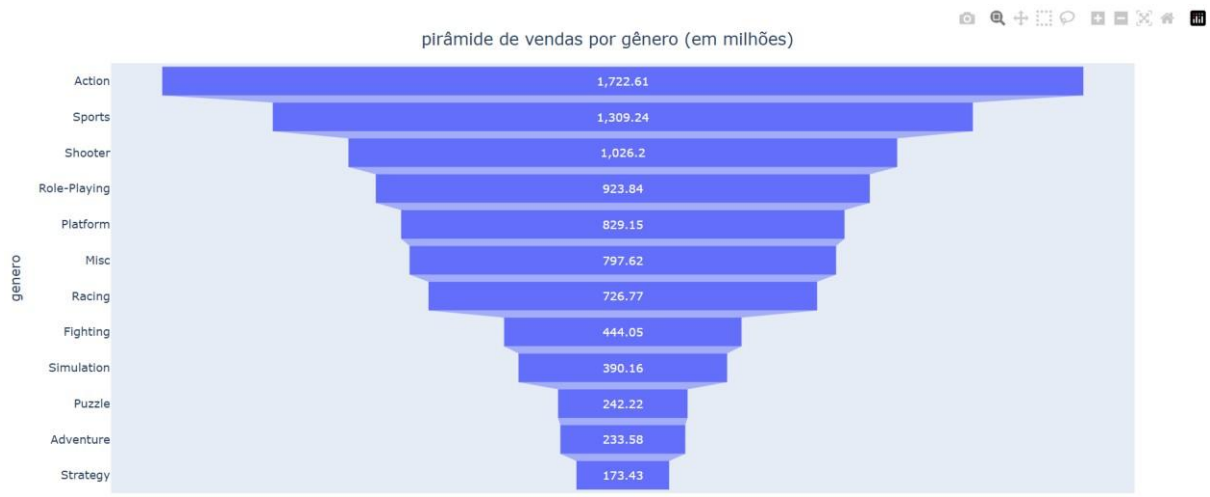
```
plt.figure(figsize=(10,7))
sns.histplot(data=df, x='ano de lancamento', bins=35, kde=True)
plt.ylabel('Lançamentos', size=13)
plt.xlabel('Ano', size=13)
plt.title('Frequência anual de lançamentos', size=15)
plt.yticks(fontsize=12)
plt.xticks(fontsize=12, rotation=80)
plt.show()
```

4. Pirâmide de Vendas por Gênero

O funil gerado com Plotly mostra as vendas totais agrupadas por gênero de jogo, destacando os estilos mais consumidos. A visualização em funil ajuda a entender a fatia de mercado de cada gênero.

Gêneros com maiores vendas indicam as preferências dominantes dos jogadores e tendências de mercado.

Essa análise é útil para desenvolvedores e publicadoras definirem estratégias focadas.



Registrado abaixo o código fonte utilizado para gerar o gráfico acima:

```
genres = df.groupby('genero')['vendas
totais'].sum().sort_values(ascending=False).reset_index()
```

```
fig = px.funnel(genres, y='genero', x='vendas totais')


fig.update_layout(
    title={
        'text': "pirâmide de vendas por gênero (em milhões)",
        'y': 0.95,
        'x': 0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    }
)
fig.show()
```

Relatório de Análise e Classificação de Jogos com KNN

O objetivo deste experimento foi aplicar um modelo de *Machine Learning* (Aprendizado de Máquina) para tentar prever o gênero de um jogo eletrônico com base nas vendas por região geográfica.
A base de dados utilizada (vgsales(in).csv) contém informações históricas sobre jogos, incluindo vendas em milhões de unidades na América do Norte, Europa, Japão e outras regiões do mundo.

Antes da aplicação do modelo KNN, foram realizadas as seguintes etapas de pré-processamento:

- Renomeação das colunas para português, facilitando a compreensão e documentação.
- Remoção de linhas sem o valor do ano de lançamento.
- Substituição de valores ausentes na coluna *publicadora* por "desconhecida".
- Conversão do tipo de dado da coluna *ano de lançamento* para inteiro.
- Filtragem dos registros sem gênero definido ou cujas vendas somadas eram zero — evitando assim dados irrelevantes ou nulos para o modelo.

As variáveis numéricas utilizadas como entradas (features) foram:

- vendas américa do norte

- vendas europa
- vendas japao
- vendas outras regioes

A variável de saída (target) foi o gênero do jogo.

Como o modelo KNN (K-Nearest Neighbors) trabalha apenas com valores numéricos:

- Os gêneros, originalmente em texto, foram codificados em números usando LabelEncoder().
- As colunas de vendas foram padronizadas (escalonadas) com StandardScaler() para garantir que todas as variáveis tivessem o mesmo peso na distância euclidiana, já que o KNN é sensível à escala dos dados.

O conjunto foi dividido em:

- 80% para treinamento do modelo;
- 20% para teste, garantindo uma avaliação imparcial do desempenho do modelo.

A divisão foi feita de forma estratificada, mantendo a proporção original dos gêneros.

Foi aplicado o algoritmo KNN (K-Nearest Neighbors) com os seguintes parâmetros:

- n_neighbors = 5 → cada previsão considera os 5 vizinhos mais próximos;
- metric = 'minkowski' e p = 2 → que corresponde à distância euclidiana.

O modelo foi então treinado com os dados escalonados.

Após o treinamento, o modelo foi avaliado sobre o conjunto de teste.
As métricas de avaliação utilizadas foram:

- Acurácia (Accuracy): percentual de previsões corretas.
- Precisão (Precision): proporção de previsões corretas entre os casos classificados como positivos.

- Revocação (Recall): proporção de acertos entre os casos que realmente pertencem àquela classe.
- F1-score: média harmônica entre precisão e revocação.

Registrado abaixo o código fonte utilizado para o relatório KNN:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import pandas as pd

# 1) Features e alvo
X = df[['vendas américa do norte', 'vendas europa', 'vendas japao', 'vendas
outras regioes']].copy()
y = df['genero'].copy()

# 2) Limpeza
mask_valid = y.notna() & (X.sum(axis=1) > 0)
X = X[mask_valid]
y = y[mask_valid]

# 3) Codificação dos gêneros
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# 4) Treino e teste
X_train, X_test, y_train, y_test = train_test_split(
    X, y_encoded, test_size=0.20, random_state=42, stratify=y_encoded
)

# 5) Escalonamento
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 6) Modelo KNN
knn = KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2)
knn.fit(X_train_scaled, y_train)

# 7) Predição e avaliação
```

```
y_pred = knn.predict(X_test_scaled)

print("\n=== RESULTADOS KNN (vendas por região -> prever gênero) ===")
print("Número de amostras (treino):", X_train.shape[0], "| (teste):",
X_test.shape[0])
print("Acurácia no conjunto de teste: {:.2f}%".format(accuracy_score(y_test,
y_pred) * 100))

# 8) Relatório em português
relatorio = classification_report(
    y_test, y_pred, target_names=le.classes_, output_dict=True
)
df_relatorio = pd.DataFrame(relatorio).transpose()
df_relatorio.rename(columns={
    'precision': 'precisao',
    'recall': 'revocacao',
    'f1-score': 'f1_score',
    'support': 'suporte'
}, inplace=True)

print("\nRelatório de classificação (em português):\n")
print(df_relatorio.round(2))

# 9) Matriz de confusão com nomes dos gêneros
cm = confusion_matrix(y_test, y_pred)
cm_df = pd.DataFrame(cm, index=le.classes_, columns=le.classes_)

print("\nMatriz de Confusão (linhas = verdadeiro, colunas = previsto):\n")
print(cm_df)

# 10) Exemplo de predição para um novo jogo
novo_jogo = np.array([[1.2, 0.8, 0.3, 0.1]])
novo_jogo_scaled = scaler.transform(novo_jogo)
genero_previsto = le.inverse_transform(knn.predict(novo_jogo_scaled))
print("\nExemplo: gênero previsto para vendas [AN=1.2, EU=0.8, JP=0.3,
Outras=0.1] ->", genero_previsto[0])
```

Essas métricas foram apresentadas em formato de tabela, já traduzidas para o português.

Também foi gerada uma matriz de confusão exibindo as classificações corretas e incorretas para cada gênero — com linhas representando os valores reais e colunas representando as previsões do modelo.

A análise demonstrou que é possível utilizar o algoritmo KNN para prever o gênero de um jogo com base nas vendas regionais, obtendo resultados coerentes com a distribuição dos dados.
Esse tipo de modelagem pode ser útil para identificar padrões de consumo por região, prever tendências de mercado e auxiliar em decisões estratégicas de desenvolvimento ou marketing no setor de jogos.

1) Novos Imports

Os novos imports foram adicionados no início do arquivo para organizar melhor o código e garantir que todas as bibliotecas necessárias tanto para o KNN quanto para o modelo de regressão estejam carregadas antes da execução. Essa padronização é uma boa prática, pois evita erros de importação no meio do código e facilita a visualização de todas as dependências utilizadas no projeto.

Os imports incluem:

- bibliotecas para manipulação de dados (Pandas, NumPy);
- bibliotecas de visualização (Matplotlib, Seaborn, Plotly);
- ferramentas de pré-processamento (StandardScaler e LabelEncoder);
- algoritmos de Machine Learning (KNN e RandomForestRegressor);
- métricas de avaliação (classificação e regressão).

Com isso, o projeto fica organizado, limpo e de fácil leitura.

```
import os
from matplotlib.backends.backend_pdf import PdfPages
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

2) Atualização do bloco do KNN utilizando nomes de colunas

O bloco original do KNN foi substituído por uma versão mais clara e estruturada, usando nomes de colunas ao invés de índices numéricos.
Isso torna o código mais compreensível e reduz o risco de erros caso a ordem das colunas mude no dataset.

No novo formato:

- As features (entradas do modelo) são explicitamente selecionadas pelos nomes das colunas:
  vendas américa do norte, *vendas europa*, *vendas japão* e *vendas outras regiões*.
- O alvo (saída) é a coluna gênero, também chamada pelo nome.
- Foi mantido o processo essencial de pré-processamento:
  - remoção de linhas inválidas,
  - codificação dos gêneros via LabelEncoder,
  - padronização dos dados com StandardScaler,
  - divisão entre treino e teste mantendo a proporção das classes (stratify).

Essa nova abordagem deixa o código mais intuitivo e alinhado com boas práticas de programação em ciência de dados.

```
# 10) Exemplo de predição para um novo jogo
novo_jogo = pd.DataFrame([[1.2, 0.8, 0.3, 0.1]], columns=X.columns)
novo_jogo_scaled = scaler.transform(novo_jogo)
genero_previsto = le.inverse_transform(knn.predict(novo_jogo_scaled))
print("\nExemplo: gênero previsto para vendas [AN=1.2, EU=0.8, JP=0.3, Outras=0.1] ->", genero_previsto[0])
```

3) Inserção do bloco de Regressão (RandomForestRegressor)

Após o KNN, foi inserido um segundo modelo baseado em regressão usando a técnica Random Forest Regressor.

Enquanto o KNN é utilizado para classificação (prever o gênero do jogo),
o RandomForestRegressor é aplicado para prever um valor numérico, neste caso:

a previsão das vendas totais de um jogo com base nas vendas por região.

O bloco inclui:

• Preparação dos dados

Foram selecionadas as mesmas colunas de vendas regionais como variáveis de entrada (X_reg)
e a coluna *vendas totais* como variável alvo (y_reg).

Linhas com valores faltantes são eliminadas para garantir que o modelo não receba dados inválidos.

• Separação dos dados

Assim como no KNN, os dados são divididos em treino (80%) e teste (20%).

• Escalonamento

Apesar de florestas aleatórias não exigirem escalonamento, ele foi mantido para consistência com o modelo anterior.

• Treinamento do modelo

O RandomForestRegressor cria diversas árvores de decisão que, juntas, produzem uma previsão mais estável e robusta.

• Avaliação do modelo

São calculadas métricas importantes:

- MSE (Mean Squared Error)
- RMSE (Root Mean Squared Error)
- R^2 (coeficiente de determinação)

Essas métricas permitem avaliar o quanto o modelo explica os valores reais de vendas.

• Predição de exemplo

Assim como no KNN, foi adicionado um exemplo de previsão usando valores fictícios de vendas por região.
Esse exemplo ajuda na interpretação prática do resultado.

```
# ------------------ INICIO: BLOCO REGRESSAO ------------------
# Objetivo: prever vendas totais (em milhoes) usando ano, console, genero, publicadora.
# Modelo: RandomForestRegressor + one-hot para categorias.

features_reg = ['ano de lancamento', 'console', 'genero', 'publicadora']
target_reg = 'vendas totais'
df_reg = df[features_reg + [target_reg]].copy()
df_reg = df_reg.dropna(subset=[target_reg])
df_reg = df_reg[df_reg[target_reg] > 0]

X_reg = df_reg[features_reg]
y_reg = df_reg[target_reg]

X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(
    X_reg, y_reg, test_size=0.20, random_state=42
)
```

```
colunas_categoricas = ['console', 'genero', 'publicadora']
colunas_numericas = ['ano de lancamento']

preprocessador = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), colunas_categoricas),
        ('num', 'passthrough', colunas_numericas)
    ]
)

modelo_reg = RandomForestRegressor(
    n_estimators=300, random_state=42, n_jobs=-1, min_samples_leaf=2
)

pipeline_reg = Pipeline(steps=[
    ('preprocessamento', preprocessador),
    ('modelo', modelo_reg)
])
```

```
pipeline_reg.fit(X_train_reg, y_train_reg)
y_pred_reg = pipeline_reg.predict(X_test_reg)

mae = mean_absolute_error(y_test_reg, y_pred_reg)
mse = mean_squared_error(y_test_reg, y_pred_reg)
rmse = np.sqrt(mse)
r2 = r2_score(y_test_reg, y_pred_reg)

print("\n=== RESULTADOS REGRESSAO (prever vendas totais) ===")
print("Amostras -> treino:", X_train_reg.shape[0], "| teste:", X_test_reg.shape[0])
print("MAE: {:.3f} milhoes".format(mae))
print("RMSE: {:.3f} milhoes".format(rmse))
print("R2: {:.3f}".format(r2))

exemplo_reg = pd.DataFrame([{
    'ano de lancamento': 2016,
    'console': 'PS4',
    'genero': 'Action',
    'publicadora': 'Sony Computer Entertainment'
}])
prev_exemplo = pipeline_reg.predict(exemplo_reg)[0]
print("\nExemplo: vendas previstas para jogo (PS4, Action, 2016, Sony) -> {:.3f} milhoes".format(prev_exemplo))
# ------------------ FIM: BLOCO REGRESSAO ------------------
```

Anderson e Guilherme Marconi