FATEC DE REGISTRO

DESENVOLVIMENTO DE SOFTWARE MULTIPLATAFORMA

ADELDIVO ALVES DE SOUSA JUNIOR

TRABALHO DE MINERAÇÃO DE DADOS

PROCESSO KDD

Registro

2024

SUMÁRIO

1. INTRODUÇÃO E CONCEITOS

A mineração de dados, uma área interdisciplinar que combina estatística, inteligência artificial e aprendizado de máquina, foca na extração de padrões úteis e informações ocultas em grandes volumes de dados. Esse processo está intimamente ligado ao *KDD* (*Knowledge Discovery in Databases*), que se refere a uma metodologia estruturada para descoberta de conhecimento em bases de dados.

1.1. O Processo KDD

O processo *KDD* é composto por cinco etapas principais:

1. **Seleção:** Identificação de dados relevantes para o problema em análise.
2. **Pré-processamento:** Limpeza e organização para eliminar inconsistências ou valores ausentes.
3. **Transformação:** Ajuste dos dados para formatos apropriados à aplicação de algoritmos analíticos.
4. **Mineração de Dados:** Aplicação de técnicas como classificação, agrupamento, associação e predição para identificar padrões.
5. **Interpretação/Avaliação:** Validação e contextualização dos resultados para gerar conhecimento útil.

1.2. Diferenças entre Mineração de Dados e Ferramentas Tradicionais

Enquanto ferramentas tradicionais de análise exploram estatísticas descritivas e consultas predefinidas, a mineração de dados utiliza algoritmos avançados para identificar padrões ocultos e prever comportamentos futuros, permitindo insights mais profundos e acionáveis.

2. IMPLEMENTAÇÃO PRÁTICA

Nesta seção, é apresentado um exemplo prático do Processo KDD seguindo as principais etapas mencionadas anteriormente.

2.1. Seleção

Para fins ilustrativos, será um conjunto de dados de registros de vendas de um *e-commerce*, incluindo informações como ID do cliente, categoria de produto, quantidade comprada, data da compra, valor total e se o cliente é VIP. Esses dados podem ser obtidos de sistemas CRM ou ERP.

Quadro 1. Base de dados de vendas.

ID_Cliente	Categoria_Produto	Quantidade	Data_Compra	Valor_Total (R$)	Cliente_VIP
101	Eletrônicos	2	2024-01-10	1500.00	Sim
102	Eletrodomésticos	1	2024-01-12	700.00	Não
103	Móveis	3	2024-01-15	2100.00	Sim
104	Vestuário	5	2024-01-18	500.00	Não
105	Eletrônicos	1	2024-01-20	800.00	Não
106	Eletrodomésticos	2	2024-01-22	1400.00	Sim
107	Móveis	1	2024-01-25	700.00	Não
108	Vestuário	4	2024-01-28	400.00	Não
109	Eletrônicos	3	2024-01-30	2400.00	Sim
110	Eletrodomésticos	1	2024-02-01	600.00	Não

Fonte: Autoria própria, 2024.

2.2. Pré-processamento

Para garantir que os dados estejam consistentes e prontos para análise, utilizamos as bibliotecas *pandas* e *numpy* para tratar valores ausentes e corrigir inconsistências.

O código abaixo demonstra como isso foi realizado:

Figura 1. Pré-processamento.

```python
import pandas as pd

# Carregar dados
dados = pd.read_csv('db/vendas.csv')

# Remover valores ausentes
dados = dados.dropna()

# Conversão de formatos inconsistentes
dados['Data_Compra'] = pd.to_datetime(dados['Data_Compra'])
```

Fonte: Autoria própria, 2024.

2.3. Transformação

Os dados foram convertidos para formatos que permitem a aplicação de algoritmos. Por exemplo, valores categóricos de texto foram transformados em números usando o *LabelEncoder*,

preparando-os para algoritmos de mineração de dados.

Segue o código para essa etapa:

Figura 2. Transformação.

```python
from sklearn.preprocessing import LabelEncoder

# Aplicar Label Encoding na categoria do produto, mantendo a coluna original para o gráfico
encoder = LabelEncoder()
dados['Categoria_Produto_Cod'] = encoder.fit_transform(dados['Categoria_Produto'])
```

Fonte: Autoria própria, 2024.

2.4. Mineração de Dados

Aplicação de algoritmos de classificação para segmentar clientes com base em padrões de compra. Para o exemplo, foi utilizado o algoritmo de árvore de decisão.

Figura 3. Classificação.

```python
from sklearn.tree import DecisionTreeClassifier

# Definir as variáveis independentes (X) e alvo (y)
X = dados[['Quantidade', 'Categoria_Produto_Cod']]
y = dados['Cliente_VIP'].apply(lambda x: 1 if x == 'Sim' else 0)  # Convertendo para binário

# Treinamento do modelo
modelo = DecisionTreeClassifier()
modelo.fit(X, y)

# Exemplo de predição
predicao = modelo.predict([[2, 1]])  # Exemplo com quantidade 2 e categoria codificada como 1
print("Predição do modelo:", predicao)
```

Fonte: Autoria própria, 2024.

2.5. Clusterização

Além da classificação, foi aplicado o algoritmo *k-means* para segmentar clientes com base em padrões de comportamento. Clusterização é uma técnica de aprendizado de máquina não supervisionada que agrupa dados em subconjuntos ou *clusters*, de modo que os dados dentro de cada cluster sejam mais semelhantes entre si do que aos dados em outros clusters. Este método é útil para identificar padrões ou segmentos dentro de um conjunto de dados, permitindo uma análise mais detalhada e a criação de estratégias específicas para cada grupo.

2.5.1. Algoritmo *k-means*

O *k-means* é um dos algoritmos de clusterização mais populares e amplamente utilizados. O objetivo do *k-means* é dividir um conjunto de *n* observações em *k clusters*, onde cada observação pertence ao cluster cujo centro (chamado de *centróide*) está mais próximo. O processo funciona da seguinte maneira:

1. **Inicialização**: Escolhe-se inicialmente *k* pontos aleatórios como *centróides*.
2. **Atribuição**: Cada ponto de dados é atribuído ao *centróide* mais próximo, formando *k clusters*.
3. **Atualização**: Calcula-se a média dos pontos em cada cluster para encontrar novos *centróides*.
4. **Repetição**: Os passos de atribuição e atualização são repetidos até que os *centróides* não mudem significativamente ou um número máximo de iterações seja alcançado.

A Figura 4 apresenta a aplicação prática do algoritmo *k-means* em nosso conjunto de dados para segmentar clientes com base em padrões de comportamento.

Figura 4. Clusterização.

```python
from sklearn.cluster import KMeans

# Aplicação do algoritmo k-means
kmeans = KMeans(n_clusters=3, random_state=42)
dados['Cluster'] = kmeans.fit_predict(X)

# Visualização dos clusters
print(dados[['ID_Cliente', 'Cluster']])
```

Fonte: Autoria própria, 2024.

Essa abordagem permitiu identificar três segmentos distintos de clientes, baseando-se em suas características de compra. Cada cliente foi atribuído a um dos *clusters*, o que, na prática, proporciona *insights* valiosos para personalizar estratégias de *marketing* e melhorar a experiência do cliente.

2.6. Associação

Para identificar produtos frequentemente comprados juntos, foram utilizadas regras de associação com o algoritmo *fpgrowth*. A associação é uma técnica de mineração de dados que identifica padrões recorrentes, correlações ou estruturas causais nos dados. Utilizando esta técnica,

podemos descobrir conjuntos de itens que aparecem frequentemente juntos nas transações.

Figura 5. Associação.

```python
from mlxtend.frequent_patterns import fpgrowth

# Preparar os dados para análise de associação
dados_crosstab = pd.crosstab(dados['ID_Cliente'], dados['Categoria_Produto_Cod']).astype(bool)

# Encontrar itens frequentes usando fpgrowth
frequent_itemsets = fpgrowth(dados_crosstab, min_support=0.2, use_colnames=True)
print(frequent_itemsets)
```

Fonte: Autoria própria, 2024.

Neste exemplo, o algoritmo *fpgrowth* foi usado para identificar os conjuntos de itens frequentes com um suporte mínimo de 20%, facilitando a descoberta de produtos que são frequentemente comprados juntos.

2.7. Interpretação/Avaliação

Os resultados foram avaliados para entender padrões e tendências nas vendas, permitindo uma análise detalhada do desempenho de diferentes categorias de produtos. Utilizando visualizações, como gráficos de barras, é possível identificar quais categorias de produtos geram mais receita e quais precisam de mais atenção.

Figura 6. Gráfico de Receita Total por Categoria de Produto (Código).

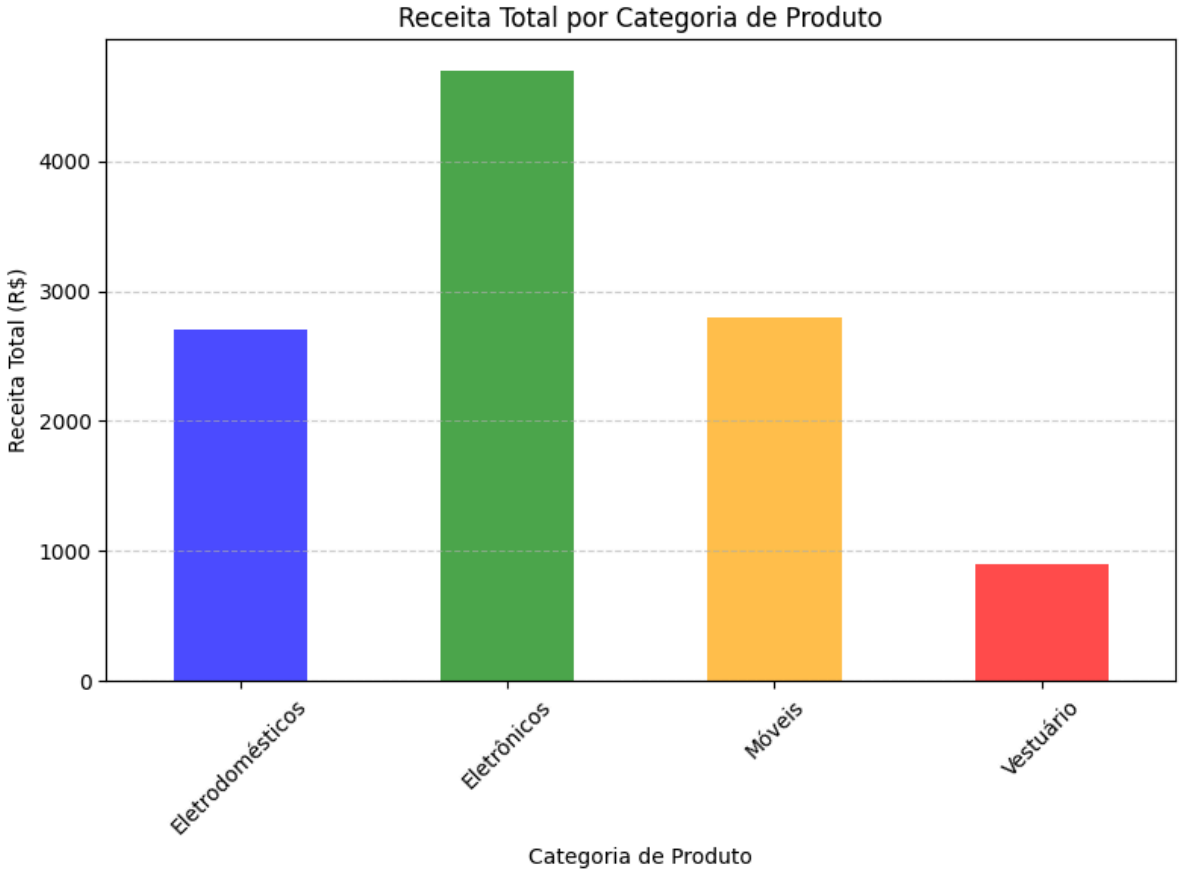
```python
import matplotlib.pyplot as plt

# Agrupar os dados por Categoria_Produto e somar o Valor_Total
receita_por_categoria = dados.groupby('Categoria_Produto')['Valor_Total'].sum()

# Gerar o gráfico de barras
plt.figure(figsize=(8, 6))
receita_por_categoria.plot(kind='bar', color=['blue', 'green', 'orange', 'red'], alpha=0.7)
plt.title('Receita Total por Categoria de Produto')
plt.xlabel('Categoria de Produto')
plt.ylabel('Receita Total (R$)')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()

plt.show()

```

Fonte: Autoria própria, 2024.

O gráfico de barras abaixo (Gráfico 1) apresenta a receita total gerada por cada categoria de produto. As categorias são diferenciadas por cores, facilitando a comparação visual. A partir desse

gráfico, podemos observar que a categoria "Eletrônicos" gerou a maior receita, enquanto "Vestuário" gerou a menor receita. Essas informações são cruciais para orientar decisões estratégicas, como alocação de recursos, estratégias de *marketing* e ajustes no portfólio de produtos.

Gráfico 1. Receita Total por Categoria de Produto.



Fonte: Autoria própria, 2024.

3. ANÁLISE CRÍTICA

O processo de *KDD* apresenta um conjunto de vantagens significativas que o tornam essencial em diferentes setores. Ele facilita a identificação de padrões ocultos nos dados, contribuindo diretamente para a tomada de decisão informada. Por exemplo, pode ser utilizado para prever tendências de mercado, identificar riscos financeiros ou personalizar ofertas de produtos para clientes específicos. Sua aplicabilidade é notável em áreas tão distintas quanto saúde, para análises de diagnósticos, e segurança, para a prevenção de fraudes. A capacidade de ser adaptado a diversos contextos confere ao *KDD* uma relevância estratégica no cenário moderno de análise de dados.

No entanto, o sucesso do *KDD* depende fortemente da qualidade dos dados utilizados. Dados inconsistentes ou incompletos podem comprometer a eficácia dos algoritmos e gerar conclusões equivocadas. Além disso, as exigências computacionais de alguns algoritmos avançados podem ser

um desafio, especialmente para organizações com infraestrutura limitada. Outro ponto de atenção é que, embora os resultados gerados sejam poderosos, eles frequentemente demandam interpretação cuidadosa por especialistas para garantir que os insights sejam aplicáveis e façam sentido no contexto do negócio.

Um exemplo prático da aplicação do *KDD* é a análise preditiva em finanças, como a previsão de inadimplências com base no histórico de clientes. Esse tipo de análise pode ajudar bancos e instituições financeiras a mitigar riscos. Contudo, desafios surgem ao lidar com dados dinâmicos, como aqueles provenientes de redes sociais, que frequentemente mudam em tempo real e requerem sistemas mais complexos para integração e processamento.

Por fim, a avaliação humana permanece indispensável no ciclo do *KDD*. Apesar do alto nível de automação, é o contexto específico e o conhecimento do domínio que determinam a utilidade dos insights gerados. Dessa forma, o *KDD* não substitui a expertise humana, mas a complementa, fornecendo um ferramental robusto para análise e tomada de decisão baseada em dados.

4. CONCLUSÃO

O processo *KDD* é uma ferramenta poderosa para a descoberta de conhecimento em grandes volumes de dados, com aplicabilidade em diversos contextos. No entanto, seu sucesso depende de dados de alta qualidade, algoritmos eficientes e da avaliação humana dos resultados. Apesar das limitações, suas vantagens tornam-no essencial em áreas como negócios, saúde e segurança.

5. REFERÊNCIAS

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. *From Data Mining to Knowledge Discovery in Databases*. *AI Magazine*, 1996.

HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. Elsevier, 2011.

ZAKI, M. J.; MEIRA, W. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, 2014.

KDD CUP. *Knowledge Discovery and Data Mining Competition Resources*. Disponível em: <http://www.kdd.org/kdd-cup/>. Acesso em: 30 nov. 2024.

PANDAS. *Pandas Documentation*. Disponível em: <https://pandas.pydata.org/pandas-docs/stable/ />. Acesso em: 30 nov. 2024.

SCIKIT-LEARN. *Scikit-learn: Machine Learning in Python*. Disponível em: <https://scikit-learn.org/stable/ />. Acesso em: 30 nov. 2024.

MLXTEND. *mlxtend: Machine Learning Extensions*. Disponível em: <https://rasbt.github.io/mlxtend/ />. Acesso em: 30 nov. 2024.

MATPLOTLIB. *Matplotlib Documentation*. Disponível em: <https://matplotlib.org/stable/contents.html />. Acesso em: 30 nov. 2024.