Mineração de Dados: Um Guia Completo

Introdução à Mineração de Dados

A Mineração de Dados é um campo interdisciplinar que se concentra na descoberta de padrões, tendências e informações úteis a partir de grandes conjuntos de dados. Ela combina técnicas de estatística, aprendizado de máquina e sistemas de banco de dados para transformar dados brutos em conhecimento acionável. O objetivo principal é extrair insights que possam apoiar a tomada de decisões em diversas áreas, como negócios, ciência e engenharia.

Relação com Inteligência Artificial e Aprendizado de Máquina

A Mineração de Dados está intrinsecamente ligada à Inteligência Artificial (IA) e ao Aprendizado de Máquina (ML), mas possui suas próprias distinções:

- **Inteligência Artificial (IA)**: É o conceito mais amplo, que engloba a criação de sistemas capazes de realizar tarefas que normalmente exigiriam inteligência humana, como reconhecimento de padrões, raciocínio lógico e tomada de decisão [Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach. Pearson.].
- **Aprendizado de Máquina (ML)**: É um subcampo da IA que estuda algoritmos que permitem aos computadores aprender a partir de dados, sem serem explicitamente programados. Um sistema de ML melhora seu desempenho em uma tarefa com a experiência adquirida [Mitchell, T. M. (1997). Machine Learning. McGraw-Hill].
- **Mineração de Dados (MD)**: Utiliza técnicas de Aprendizado de Máquina e estatística para descobrir padrões úteis em grandes bases de dados. Embora o ML seja uma ferramenta poderosa na MD, a MD não se limita apenas a predições, buscando também a descoberta de padrões e o entendimento dos dados [Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques. Elsevier.].

Em resumo, a IA é o campo geral, o ML é uma de suas abordagens para permitir que máquinas aprendam, e a Mineração de Dados aplica essas e outras técnicas para extrair conhecimento de grandes volumes de dados. [MD-AULA03-NOMENCLATURAS.pdf, página 7]

O Processo de Descoberta de Conhecimento em Dados (KDD)

O KDD (Knowledge Discovery in Databases) é um processo iterativo e interativo que visa extrair conhecimento de alto nível a partir de dados de baixo nível. O processo é composto por uma sequência de etapas, cada uma com seus próprios objetivos e desafios, que transformam dados brutos em conhecimento acionável.

1. **Seleção dos Dados (Data Selection)**: A primeira etapa do processo KDD é a seleção de um subconjunto de dados de um grande repositório de dados. O objetivo é criar um conjunto de dados alvo que seja relevante para a análise e o problema de negócio. Esta etapa envolve a compreensão do domínio do problema, a identificação das fontes de dados relevantes e a extração dos dados necessários. A qualidade da seleção de dados impacta diretamente a qualidade dos resultados da mineração de dados.

2. **Pré-processamento e Limpeza dos Dados (Data Preprocessing and Cleaning)**: Os dados do mundo real são frequentemente incompletos, inconsistentes e ruidosos. A etapa de pré-processamento e limpeza de dados visa tratar esses problemas. As principais tarefas incluem o tratamento de valores ausentes (imputação, remoção), a suavização de dados ruidosos (binning, regressão) e a correção de inconsistências. O objetivo é garantir que os dados sejam precisos, completos e consistentes, o que é crucial para a eficácia dos algoritmos de mineração de dados.

3. **Transformação dos Dados (Data Transformation)**: Nesta etapa, os dados pré-processados são transformados em um formato adequado para a mineração. As técnicas de transformação de dados incluem a normalização (escalonamento dos dados para um intervalo específico), a padronização (transformação dos dados para ter média zero e desvio padrão um), a discretização (conversão de atributos contínuos em atributos categóricos) e a criação de novos atributos

(engenharia de atributos). A transformação de dados pode melhorar a precisão e a eficiência dos algoritmos de mineração de dados.

4. **Mineração de Dados (Data Mining)**: Esta é a etapa central do processo KDD, onde algoritmos inteligentes são aplicados para extrair padrões de dados. A escolha do algoritmo de mineração de dados depende do tipo de padrão a ser descoberto. As principais tarefas da mineração de dados incluem a classificação (atribuição de itens a classes predefinidas), a regressão (previsão de um valor contínuo), o clustering (agrupamento de itens semelhantes), a análise de associação (descoberta de regras de associação) e a detecção de anomalias (identificação de itens, eventos ou observações que não se conformam com um padrão esperado).

5. **Interpretação e Avaliação dos Padrões (Pattern Interpretation and Evaluation)**: A etapa final do processo KDD é a interpretação e avaliação dos padrões descobertos. Nem todos os padrões descobertos são necessariamente úteis ou interessantes. Nesta etapa, os padrões são avaliados com base em medidas de interesse, como precisão, suporte e confiança. A visualização de dados pode ser usada para ajudar na interpretação dos padrões. O conhecimento extraído é então apresentado aos usuários de uma forma compreensível e acionável, que pode ser usado para apoiar a tomada de decisões.

Pré-processamento de Dados

O pré-processamento de dados é uma etapa fundamental no processo de KDD, responsável por preparar os dados brutos para que possam ser utilizados de forma eficiente pelos algoritmos de mineração. Dados de baixa qualidade podem levar a resultados de mineração de baixa qualidade, invalidando toda a análise. Portanto, é crucial tratar problemas como dados incompletos, inconsistentes e ruidosos.

Principais Problemas com Dados Brutos

Os dados brutos, também conhecidos como dados de origem ou atômicos, frequentemente apresentam os seguintes problemas:

- **Incompletude**: Ocorre quando faltam valores em um ou mais atributos. Por exemplo, um cliente pode não ter preenchido o campo de renda mensal em um

formulário de cadastro.

- **Inconsistência**: Acontece quando há informações conflitantes nos dados. Por exemplo, a idade de uma pessoa pode estar registrada como 150 anos, o que é improvável, ou o mesmo cliente pode ter endereços diferentes em registros distintos.
- **Ruído**: Refere-se a erros ou variações aleatórias nos dados. Isso pode ser causado por erros de digitação, falhas em sensores de coleta de dados ou informações deliberadamente falsas fornecidas pelos usuários.

Esses problemas podem inviabilizar uma análise ou levar a conclusões equivocadas. Portanto, o pré-processing é essencial para garantir a qualidade e a confiabilidade dos resultados da mineração de dados. [MD-AULA03-PRE-PROCESSAMENTO.pdf, páginas 3-6]

Limpeza de Dados

A limpeza de dados é a primeira sub-etapa do pré-processamento e foca em lidar com valores ausentes, dados ruidosos e inconsistências. [MD-AULA03-PRE-PROCESSAMENTO.pdf, página 18]

Tratamento de Valores Ausentes

Valores ausentes são comuns e podem ocorrer por diversas razões. O tratamento adequado é crucial para não enviesar a base de dados. Algumas estratégias incluem:

- **Exclusão**: Remover registros ou colunas que contêm valores ausentes. Esta abordagem é simples, mas pode resultar na perda de informações valiosas, especialmente se a porcentagem de dados ausentes for alta. [MD-AULA03-PRE-PROCESSAMENTO.pdf, página 19]
- **Imputação Estatística**: Preencher valores ausentes com medidas de tendência central, como média, mediana ou moda. A escolha da medida depende da distribuição dos dados (média para distribuições simétricas, mediana para distribuições assimétricas). [MD-AULA03-PRE-PROCESSAMENTO.pdf, página 19]
- **Criação de Categoria 'Desconhecido'**: Para variáveis categóricas, uma nova categoria pode ser criada para representar os valores ausentes. [MD-AULA03-PRE-PROCESSAMENTO.pdf, página 19]

- **Interpolação ou Modelos Preditivos**: Estimar valores ausentes usando técnicas mais avançadas, como interpolação (para dados sequenciais) ou modelos de aprendizado de máquina. [MD-AULA03-PRE-PROCESSAMENTO.pdf, página 19]
- **Algoritmos que Lidam com Ausências**: Alguns algoritmos de aprendizado de máquina são capazes de trabalhar diretamente com dados incompletos, sem a necessidade de imputação prévia. [MD-AULA03-PRE-PROCESSAMENTO.pdf, página 19]

Tratamento de Dados Ruidosos

Ruído refere-se a erros ou variações aleatórias nos dados. Técnicas para suavizar dados ruidosos incluem:

- **Binning (Agrupamento)**: Dividir os dados em 'caixas' ou 'bins' e, em seguida, suavizar os valores dentro de cada bin, substituindo-os pela média, mediana ou limites do bin. [Data-Preprocessing.pdf, página 7]
- **Regressão**: Ajustar os dados a uma função matemática (e.g., regressão linear) para modelar a relação entre variáveis e, assim, suavizar o ruído. [Data-Preprocessing.pdf, página 8]
- **Análise de Outliers**: Identificar e tratar valores atípicos que se desviam significativamente do restante dos dados. Outliers podem ser ruído ou indicar informações importantes. [Data-Preprocessing.pdf, página 8]

Resolução de Inconsistências

Inconsistências podem surgir de diversas fontes, como erros de entrada de dados, representações inconsistentes ou integração de dados de múltiplas fontes. A detecção de discrepâncias e a aplicação de transformações para corrigi-las são essenciais. Ferramentas de limpeza de dados e auditoria de dados podem auxiliar nesse processo. [Data-Preprocessing.pdf, página 9-10]

Integração de Dados

A integração de dados é o processo de combinar dados de múltiplas fontes heterogêneas em um armazenamento de dados coerente. Uma integração cuidadosa pode reduzir redundâncias e inconsistências, melhorando a precisão e a velocidade do processo de mineração de dados subsequente. [Data-Preprocessing.pdf, página 11]

Os principais desafios na integração de dados incluem:

- **Heterogeneidade Semântica**: Diferentes fontes podem usar nomes distintos para o mesmo conceito (e.g., 'customer id' em um banco de dados e 'cust number' em outro). Resolver essas inconsistências de nomenclatura é fundamental para garantir que os dados sejam corretamente combinados. [Data-Preprocessing.pdf, página 12]
- **Detecção e Resolução de Conflitos de Valores**: Valores para o mesmo atributo podem diferir entre as fontes (e.g., 'H' e 'S' para tipo de pagamento em um sistema, e '1' e '2' em outro). É necessário definir regras para resolver esses conflitos e padronizar os valores. [Data-Preprocessing.pdf, página 12]
- **Redundância de Dados**: A integração pode introduzir dados duplicados. É importante identificar e remover essas redundâncias para evitar que o processo de mineração seja prejudicado. [Data-Preprocessing.pdf, página 12]

Metadados (dados sobre os dados), como nome, significado, tipo e faixa de valores permitidos para cada atributo, são cruciais para auxiliar na integração e evitar erros. [Data-Preprocessing.pdf, página 12]

Transformação de Dados

A transformação de dados é a etapa em que os dados são convertidos em formatos apropriados para a mineração, o que pode envolver a normalização, agregação e criação de novas variáveis. Esta fase é crucial para otimizar o desempenho dos algoritmos de mineração e garantir que os dados estejam no formato ideal para análise. [MD-AULA03-PRE-PROCESSAMENTO.pdf, página 1]

Técnicas Comuns de Transformação de Dados

Diversas técnicas são empregadas na transformação de dados, cada uma com um objetivo específico:

1. Normalização (Min-Max Scaling)

A normalização é usada para escalar os valores de um atributo para um intervalo específico, geralmente entre 0 e 1. Isso é particularmente útil quando os atributos têm

diferentes escalas, evitando que atributos com valores maiores dominem a análise. A fórmula de normalização Min-Max é:

```
X_normalizado = (X - X_min) / (X_max - X_min)
```

Objetivo: Reduzir a influência de atributos com grandes variações de escala e garantir que todos os atributos contribuam igualmente para o modelo. [MD_AULA05_ATIVIDADE(1).ipynb]

2. Padronização (Z-score Standardization)

A padronização transforma os dados para que tenham média zero e desvio padrão um. Isso é útil para algoritmos que assumem que os dados seguem uma distribuição normal ou que são sensíveis à escala dos atributos (e.g., SVMs, redes neurais).

```
X_padronizado = (X - média) / desvio_padrão
```

Objetivo: Centralizar os dados em torno de zero e padronizar a variância, facilitando a convergência de alguns algoritmos de aprendizado de máquina. [MD_AULA05_ATIVIDADE(1).ipynb]

3. Transformação Logarítmica

A transformação logarítmica é aplicada a dados com distribuições assimétricas (skewed), onde há valores muito altos que podem distorcer a análise. Ao aplicar o logaritmo, a escala dos valores é reduzida, tornando a distribuição mais simétrica.

Objetivo: Reduzir a influência de outliers e tornar a distribuição dos dados mais próxima de uma distribuição normal, o que é benéfico para muitos modelos estatísticos. [MD_AULA05_ATIVIDADE(1).ipynb]

4. Binning (Discretização)

O binning, ou discretização, é o processo de agrupar valores contínuos em intervalos (bins) ou categorias discretas. Isso pode simplificar os dados e torná-los mais fáceis de analisar, especialmente para algoritmos que trabalham melhor com dados categóricos.

Objetivo: Reduzir a complexidade dos dados contínuos, lidar com ruído e preparar dados para algoritmos que exigem entradas discretas. [Data-Cleaning-and-Preparation.pdf, página 12]

5. One-Hot Encoding

One-Hot Encoding é uma técnica para converter variáveis categóricas nominais em um formato numérico que pode ser fornecido a algoritmos de aprendizado de máquina. Para cada categoria única em uma coluna, uma nova coluna binária (0 ou 1) é criada.

Objetivo: Representar variáveis categóricas de forma que os algoritmos de aprendizado de máquina possam interpretá-las sem assumir uma ordem ou relação numérica entre as categorias. [MD_AULA05_ATIVIDADE(1).ipynb]

6. Label Encoding

Label Encoding atribui um número inteiro único a cada categoria em uma variável categórica. É adequado para variáveis ordinais, onde há uma ordem inerente entre as categorias.

Objetivo: Converter variáveis categóricas em um formato numérico para algoritmos que podem lidar com a relação ordinal implícita. [MD_AULA05_ATIVIDADE(1).ipynb]

7. Extração de Atributos (Feature Engineering)

A extração de atributos, ou engenharia de atributos, envolve a criação de novas variáveis a partir das existentes para melhorar o desempenho do modelo. Isso pode incluir a extração de componentes de datas (ano, mês, dia), a combinação de variáveis, entre outros.

Objetivo: Criar novas features que capturem melhor a informação nos dados, aumentando o poder preditivo dos modelos. [MD_AULA05_ATIVIDADE(1).ipynb]

Exemplos Práticos

Nesta seção, abordaremos a aplicação prática dos conceitos de pré-processamento e transformação de dados, bem como a construção de modelos de classificação, com foco nos princípios teóricos e nos resultados esperados, em vez de detalhes de implementação de código.

Tratamento de Duplicatas

Após a análise inicial, removemos as linhas duplicadas do dataset. A remoção de duplicatas garante que cada observação seja única, evitando vieses na análise e no

treinamento de modelos.

Tratamento de Valores Ausentes

Valores ausentes são preenchidos utilizando estratégias apropriadas, como a imputação pela média, mediana ou moda, ou a remoção de linhas/colunas, dependendo do contexto e da quantidade de dados faltantes. A escolha da técnica visa preservar a integridade dos dados e minimizar a perda de informação.

Correção de Tipos de Dados

As colunas são verificadas e convertidas para os tipos de dados corretos (e.g., numérico, categórico, data). Isso é fundamental para que os algoritmos de mineração possam processar as informações adequadamente.

Normalização (Min-Max Scaling)

Padronização (Z-score Standardization)

Binning (Discretização)

One-Hot Encoding

Label Encoding

Extração de Atributos

Extração de Atributos de Data

Vamos considerar um cenário onde temos uma coluna de datas e queremos extrair informações como ano, mês e dia. Embora o dataset `car_price_dataset.csv` não tenha uma coluna de data de registro, podemos simular a criação de uma para demonstrar a extração de atributos, ou usar a coluna 'Year' que já existe e extrair o ano dela.

Explicação Teórica:

Nesta subseção, o foco é demonstrar como atributos temporais podem ser derivados de colunas de data existentes. Por exemplo, se uma coluna contém datas completas, é possível extrair o ano, mês, dia, dia da semana, etc., como novas variáveis. Isso é útil

para capturar padrões sazonais ou tendências temporais que podem ser relevantes para o modelo. No caso de uma coluna que já representa o ano, podemos renomeá-la para maior clareza ou combiná-la com outras informações para criar atributos mais complexos.

Referências

- [1] Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach. Pearson.
- [2] Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques. Elsevier.
- [3] Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.
- [4] MD-AULA03-NOMENCLATURAS.pdf (Material fornecido)
- [5] MD-AULA02.pdf (Material fornecido)
- [6] MD-AULA03-PRE-PROCESSAMENTO.pdf (Material fornecido)
- [7] Data-Preprocessing.pdf (Material fornecido)
- [8] Data-Cleaning-and-Preparation.pdf (Material fornecido)
- [9] MD_AULA05_ATIVIDADE(1).ipynb (Material fornecido)

na(df["Price"].median(), inplace=True)`: Preenche os valores ausentes na coluna 'Price' com a mediana dos valores existentes. A mediana é uma boa escolha para dados numéricos que podem ter outliers, pois é menos sensível a eles do que a média.

Transformação de Dados em Python

Vamos ver exemplos de transformação de dados, baseados no notebook `MD_AULA05_ATIVIDADE(1).ipynb`.

Normalização (Min-Max Scaling)

A normalização Min-Max é uma técnica de reescalonamento que transforma os valores de um atributo para um intervalo específico, geralmente entre 0 e 1. Isso é feito subtraindo o valor mínimo do atributo de cada valor e dividindo pelo intervalo (máximo - mínimo). Esta técnica é útil quando os atributos têm diferentes escalas e queremos garantir que todos contribuam igualmente para o modelo.

Padronização (Z-score Standardization)

A padronização Z-score transforma os dados para que tenham uma média de 0 e um desvio padrão de 1. Isso é alcançado subtraindo a média do atributo de cada valor e dividindo pelo desvio padrão. É particularmente útil para algoritmos que assumem que os dados seguem uma distribuição normal ou que são sensíveis à escala dos atributos, como máquinas de vetores de suporte (SVMs) e redes neurais.

One-Hot Encoding

One-Hot Encoding é uma técnica para converter variáveis categóricas nominais em um formato numérico que pode ser compreendido por algoritmos de aprendizado de máquina. Para cada categoria única em uma coluna, uma nova coluna binária (0 ou 1) é criada, indicando a presença ou ausência daquela categoria. Isso evita que o modelo interprete uma ordem ou relação numérica entre categorias que não existe.

Label Encoding

Label Encoding atribui um número inteiro único a cada categoria em uma variável categórica. É mais adequado para variáveis ordinais, onde existe uma ordem inerente entre as categorias (por exemplo, 'Pequeno', 'Médio', 'Grande'). Se usado em variáveis nominais, pode levar o modelo a inferir uma ordem que não existe, o que pode ser problemático.

Engenharia de Atributos (Feature Engineering)

A engenharia de atributos é o processo de criar novas variáveis (features) a partir das existentes nos dados brutos. O objetivo é melhorar o desempenho dos modelos de aprendizado de máquina, fornecendo-lhes informações mais relevantes e de maior qualidade. Isso pode incluir a extração de componentes de datas (ano, mês, dia), a combinação de variáveis para criar novas métricas, ou a transformação de dados para capturar relações não lineares. Por exemplo, a partir de uma coluna de data, podemos extrair o ano, mês e dia como atributos separados, que podem ser mais informativos para o modelo do que a data completa.

Referências

[1] Slides da Aula 07 - Classificação em Mineração de Dados. [2] Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques (3rd ed.). Morgan Kaufmann. (Capítulo 8: Classification: Basic Concepts - Seção 8.2 Decision Tree Induction).

Exemplo Prático: Aprovação de Cartão de Crédito com Árvore de Decisão

Vamos aplicar os conceitos de Árvore de Decisão para classificar solicitações de cartão de crédito como "aprovado" ou "não aprovado" com base em dados de clientes. Utilizaremos um dataset sintético e o ambiente do Google Colab.

Configuração do Ambiente e Carregamento dos Dados

Primeiro, importamos as bibliotecas necessárias e carregamos o dataset. Certifique-se de que o arquivo `credit_card_approval.csv` esteja disponível no ambiente do Colab (faça o upload se necessário).

```
 import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

# Função para carregar o CSV de forma flexível no Colab
import os

def read_csv_flex(filename):
    if os.path.exists(filename):
        return pd.read_csv(filename)
    for base in ["/content", "/mnt/data"]:
        path = os.path.join(base, filename)
        if os.path.exists(path):
            return pd.read_csv(path)
    raise FileNotFoundError(f"Arquivo {filename} não encontrado. Envie o CSV
para o Colab e rode novamente.")

# Carregar o dataset
csv_path = "credit_card_approval.csv"
df_credit = read_csv_flex(csv_path)

print("Tamanho do dataset:", df_credit.shape)
print("Primeiras 5 linhas do dataset:")
print(df_credit.head())
print("\nInformações do dataset:")
df_credit.info()
```

Explicação: * As bibliotecas `pandas`, `numpy`, `matplotlib.pyplot` e módulos do `sklearn` são importadas. * A função `read_csv_flex` é um utilitário para carregar o CSV, adaptando-se a diferentes caminhos no ambiente do Colab. * O dataset `credit_card_approval.csv` é carregado e as primeiras linhas (`df_credit.head()`) e informações gerais (`df_credit.info()`) são exibidas para uma inspeção inicial.

Pré-processamento dos Dados

Esta etapa envolve o tratamento de valores ausentes, a codificação de variáveis categóricas e a separação dos dados em conjuntos de treinamento e teste.

```
 # Tratamento de valores ausentes
# Para colunas numéricas, imputar com a média
for col in [\"Renda_Mensal\", \"Tempo_De_Emprego\"]:
    df_credit[col].fillna(df_credit[col].mean(), inplace=True)

# Definir features (X) e target (y)
X_credit = df_credit.drop("Aprovado", axis=1)
y_credit = df_credit["Aprovado"]

# Identificar colunas categóricas e numéricas
categorical_features_credit = X_credit.select_dtypes(include=
["object"]).columns
numerical_features_credit = X_credit.select_dtypes(include=["int64",
"float64"]).columns

# Criar um pipeline de pré-processamento para variáveis categóricas e numéricas
preprocessor_credit = ColumnTransformer(
    transformers=[
        ("num", SimpleImputer(strategy=\"mean\"), numerical_features_credit),
        ("cat", OneHotEncoder(handle_unknown=\"ignore\"),
categorical_features_credit)
    ])

# Dividir os dados em conjuntos de treinamento e teste
X_train_credit, X_test_credit, y_train_credit, y_test_credit =
train_test_split(X_credit, y_credit, test_size=0.2, random_state=42,
stratify=y_credit)

print("Shape de X_train_credit após split:", X_train_credit.shape)
print("Shape de X_test_credit após split:", X_test_credit.shape)
```

Explicação: * `fillna()`: Preenche os valores ausentes nas colunas numéricas `Renda_Mensal` e `Tempo_De_Emprego` com a média dos valores existentes. * `X_credit = df_credit.drop("Aprovado", axis=1)` e `y_credit = df_credit["Aprovado"]`: Separa o dataset em features (X) e a variável alvo (y). * `ColumnTransformer` e `OneHotEncoder`: Preparam as variáveis categóricas para o modelo, convertendo-as em representações numéricas binárias. * `SimpleImputer(strategy=\"mean\")`: Garante que quaisquer valores ausentes remanescentes nas colunas numéricas sejam preenchidos com a média. * `train_test_split`: Divide o dataset em 80% para treinamento e 20% para teste, garantindo a reprodutibilidade e a estratificação da classe alvo.

Treinamento do Modelo de Árvore de Decisão

Agora, treinamos o modelo de Árvore de Decisão usando os dados pré-processados.

```
# Criar o pipeline completo: pré-processamento + modelo
model_credit = Pipeline(steps=[
    ("preprocessor", preprocessor_credit),
    ("classifier", DecisionTreeClassifier(random_state=42))
])

# Treinar o modelo
model_credit.fit(X_train_credit, y_train_credit)

print("Modelo de Árvore de Decisão para Aprovação de Cartão de Crédito treinado com sucesso!")
```

Explicação: * Um `Pipeline` é criado para encadear as etapas de pré-processamento e o classificador. * `DecisionTreeClassifier(random_state=42)`: Inicializa o classificador de Árvore de Decisão. * `model_credit.fit(X_train_credit, y_train_credit)`: Treina o modelo usando os dados de treinamento.

Avaliação do Modelo

Avaliamos o desempenho do modelo nos dados de teste para verificar sua precisão e outras métricas importantes.

```
# Fazer previsões no conjunto de teste
y_pred_credit = model_credit.predict(X_test_credit)

# Avaliar o modelo
print("Acurácia:", accuracy_score(y_test_credit, y_pred_credit))
print("\nRelatório de Classificação:\n", classification_report(y_test_credit, y_pred_credit))
print("\nMatriz de Confusão:\n", confusion_matrix(y_test_credit, y_pred_credit))
```

Explicação: * `model_credit.predict(X_test_credit)`: Realiza previsões nos dados de teste. * `accuracy_score`, `classification_report`, `confusion_matrix`: Calculam e exibem métricas de avaliação do modelo.

Visualização da Árvore de Decisão

É possível visualizar a estrutura da árvore de decisão para entender como as decisões são tomadas.

```
# Obter nomes das features após o One-Hot Encoding
feature_names_credit = model_credit.named_steps[\"preprocessor\"]\
    .transformers_[1]
[1].get_feature_names_out(categorical_features_credit).tolist()
feature_names_credit = numerical_features_credit.tolist() +
feature_names_credit

plt.figure(figsize=(25, 15))
plot_tree(model_credit.named_steps[\"classifier\"],
          feature_names=feature_names_credit,
          class_names=[\"Não Aprovado\", \"Aprovado\"],
          filled=True,
          rounded=True,
          fontsize=10)
plt.title("Árvore de Decisão para Aprovação de Cartão de Crédito")
plt.show()
```

Explicação: * `plot_tree`: Função para visualizar a árvore. `feature_names_credit` são os nomes das colunas usadas para as decisões, e `class_names` são os rótulos das classes alvo. `filled=True` colore os nós com base na classe majoritária, e `rounded=True` arredonda as caixas dos nós.

Referências

[1] Slides da Aula 07 - Classificação em Mineração de Dados. [2] Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques (3rd ed.). Morgan Kaufmann. (Capítulo 8: Classification: Basic Concepts - Seção 8.2 Decision Tree Induction).

Aprofundamento Teórico em Classificação

A classificação é uma das tarefas mais fundamentais e amplamente utilizadas na mineração de dados e no aprendizado de máquina. Seu objetivo é construir um modelo (ou classificador) que possa prever um rótulo de classe categórico para um determinado conjunto de dados de entrada. Este processo é um exemplo de **aprendizado supervisionado**, pois o algoritmo "aprende" a partir de um conjunto de dados de treinamento onde os rótulos de classe já são conhecidos.

O Processo de Classificação em Detalhes

O processo de classificação pode ser dividido em duas fases principais: **treinamento** e **teste**.

1. **Fase de Treinamento (Construção do Modelo)**: Nesta fase, um algoritmo de aprendizado de máquina é alimentado com um conjunto de dados de treinamento. Este conjunto de dados consiste em exemplos, cada um com um conjunto de atributos (features) e um rótulo de classe correspondente. O algoritmo analisa esses dados e constrói um modelo que representa o relacionamento entre os atributos e os rótulos de classe. O modelo pode assumir várias formas, como uma árvore de decisão, um conjunto de regras, uma função matemática, etc.

2. **Fase de Teste (Avaliação do Modelo)**: Após o treinamento, o modelo precisa ser avaliado para determinar sua precisão e capacidade de generalização. Para isso, um conjunto de dados de teste é utilizado. Este conjunto de dados é independente do conjunto de treinamento e também contém exemplos com rótulos de classe conhecidos. O modelo é usado para prever os rótulos de classe dos exemplos no conjunto de teste, e as previsões são comparadas com os rótulos reais. A precisão do modelo é então calculada com base no número de previsões corretas.

3. **Fase de Classificação (Uso do Modelo)**: Se a precisão do modelo for considerada aceitável, ele pode ser usado para classificar novos dados para os quais o rótulo de classe é desconhecido. Esta é a fase final, onde o modelo é implantado para resolver o problema para o qual foi projetado.

Métricas de Avaliação para Classificação

Para avaliar o desempenho de um modelo de classificação, várias métricas são utilizadas. A escolha da métrica apropriada depende do problema e da distribuição das classes.

- **Acurácia (Accuracy)**: É a métrica mais simples e representa a proporção de previsões corretas em relação ao total de previsões. No entanto, a acurácia pode ser enganosa em conjuntos de dados desbalanceados (onde uma classe é muito mais frequente que as outras).

- **Matriz de Confusão (Confusion Matrix)**: É uma tabela que resume o desempenho de um modelo de classificação. Ela mostra o número de verdadeiros positivos (TP), verdadeiros negativos (TN), falsos positivos (FP) e falsos negativos (FN).

- **Precisão (Precision)**: Mede a proporção de verdadeiros positivos entre todas as previsões positivas. É útil quando o custo de um falso positivo é alto. `Precisão = TP / (TP + FP)`
- **Recall (Sensibilidade ou Revocação)**: Mede a proporção de verdadeiros positivos que foram corretamente identificados. É útil quando o custo de um falso negativo é alto. `Recall = TP / (TP + FN)`
- **F1-Score**: É a média harmônica da precisão e do recall. É uma boa métrica para avaliar o equilíbrio entre precisão e recall. `F1-Score = 2 * (Precisão * Recall) / (Precisão + Recall)`
- **Curva ROC (Receiver Operating Characteristic)** e **AUC (Area Under the Curve)**: A curva ROC é um gráfico que mostra o desempenho de um modelo de classificação em todos os limiares de classificação. A AUC é a área sob a curva ROC e fornece uma medida agregada do desempenho do modelo em todas as classificações possíveis. Um modelo com AUC próximo de 1 é considerado excelente, enquanto um modelo com AUC próximo de 0.5 não é melhor que uma classificação aleatória.

Parâmetros e Configurações de Árvores de Decisão

Ao construir uma árvore de decisão, diversos parâmetros podem ser ajustados para controlar sua complexidade e desempenho. Estes parâmetros são cruciais para evitar o overfitting e otimizar a capacidade de generalização do modelo.

- **Critério de Divisão (Criterion)**: Define a função para medir a qualidade de uma divisão. As opções mais comuns são:
    - **Gini (Índice de Gini)**: Mede a impureza de um nó. Um valor de Gini de 0 indica um nó puro (todos os exemplos pertencem à mesma classe). O algoritmo busca minimizar o Gini. É o critério padrão para o algoritmo CART.
    - **Entropia (Entropy)**: Mede a desordem ou incerteza de um nó. Um valor de entropia de 0 indica um nó puro. O algoritmo busca maximizar o Ganho de Informação (Information Gain), que é a redução na entropia após a divisão. É o critério usado por algoritmos como ID3 e C4.5.

- **Profundidade Máxima (max_depth)**: Limita o número de níveis na árvore. Uma árvore muito profunda pode memorizar os dados de treinamento (overfitting), enquanto uma árvore muito rasa pode não capturar padrões complexos (underfitting). Definir um valor adequado ajuda a controlar a complexidade do modelo.

- **Mínimo de Amostras para Divisão (min_samples_split)**: O número mínimo de amostras que um nó deve ter para que possa ser dividido. Se um nó tiver menos amostras que este valor, ele não será dividido, tornando-se um nó folha. Isso ajuda a evitar que a árvore crie nós para pequenas variações nos dados.

- **Mínimo de Amostras por Folha (min_samples_leaf)**: O número mínimo de amostras que um nó folha deve ter. Qualquer divisão que resultaria em um nó folha com menos amostras que este valor será proibida. Isso também ajuda a suavizar o modelo e reduzir o overfitting.

- **Número Mínimo de Impureza para Divisão (min_impurity_decrease)**: Um nó será dividido se essa divisão resultar em uma diminuição de impureza maior ou igual a este valor. Se a diminuição de impureza for menor, a divisão não será realizada.

- **Balanceamento de Classes (class_weight)**: Permite atribuir pesos diferentes às classes. Isso é particularmente útil em datasets desbalanceados, onde uma classe tem muito mais exemplos que a outra. Ao dar um peso maior à classe minoritária, o modelo é incentivado a prestar mais atenção a ela, melhorando o recall para essa classe.

Podas (Pruning) e Overfitting

As árvores de decisão têm uma tendência natural a se tornarem muito complexas e a se ajustarem excessivamente aos dados de treinamento (overfitting). Isso significa que elas podem aprender o "ruído" nos dados de treinamento, resultando em um desempenho ruim em dados não vistos. Para combater o overfitting, técnicas de **poda (pruning)** são empregadas.

- **Pré-poda (Pre-pruning)**: Envolve parar o crescimento da árvore antes que ela se torne uma árvore completa. Isso é feito definindo limites para os parâmetros mencionados acima, como `max_depth`, `min_samples_split`,

`min_samples_leaf` ou `min_impurity_decrease`. A árvore para de crescer quando um desses critérios é atingido.

    - **Vantagens**: Mais eficiente computacionalmente, pois a árvore nunca é totalmente construída.
    - **Desvantagens**: Pode parar o crescimento da árvore muito cedo, perdendo padrões importantes que poderiam ser descobertos em níveis mais profundos.

- **Pós-poda (Post-pruning)**: Envolve construir a árvore de decisão completa (ou quase completa) e, em seguida, remover ou "podar" ramos que não contribuem significativamente para a precisão do modelo em um conjunto de validação. Técnicas comuns incluem a substituição de um sub-árvore por um nó folha se a remoção não aumentar o erro de validação acima de um certo limite.

    - **Vantagens**: Geralmente leva a árvores mais precisas, pois considera a árvore completa antes de podar.
    - **Desvantagens**: Mais custosa computacionalmente, pois a árvore completa é construída primeiro.

Métricas Específicas para Árvores de Decisão

Além das métricas gerais de classificação (acurácia, precisão, recall, F1-score, matriz de confusão, ROC/AUC), as árvores de decisão também podem ser avaliadas por sua **interpretabilidade** e **tamanho**.

- **Interpretabilidade**: Uma árvore menor e mais simples é geralmente mais fácil de entender e explicar. A poda contribui diretamente para isso.
- **Tamanho da Árvore**: Medido pelo número de nós (internos e folhas) ou pela profundidade máxima. Árvores menores são preferíveis se a precisão for comparável, pois são menos propensas a overfitting e mais fáceis de implantar.

Visualização de Árvores de Decisão

A visualização da estrutura da árvore é uma ferramenta poderosa para entender como o modelo toma decisões. Cada nó na visualização mostra:

- **Condição de Divisão**: O atributo e o valor de corte usados para dividir os dados (ex: `idade <= 45.5`).

- **Critério de Impureza**: O valor do Gini ou da entropia para aquele nó.
- **Amostras (samples)**: O número de exemplos que chegam a esse nó.
- **Valores (values)**: A contagem de exemplos de cada classe naquele nó.
- **Classe (class)**: A classe majoritária naquele nó, que seria a previsão se o nó fosse um nó folha.

Ferramentas como `plot_tree` do `sklearn.tree` permitem gerar representações gráficas das árvores, facilitando a interpretação do fluxo de decisão do modelo. A cor dos nós geralmente indica a classe majoritária, e a intensidade da cor pode indicar a pureza do nó.

Árvores Binárias

Uma árvore binária é uma estrutura de dados hierárquica onde cada nó tem no máximo dois filhos, geralmente referidos como filho esquerdo e filho direito. No contexto de mineração de dados e aprendizado de máquina, as árvores binárias são a base para algoritmos como as Árvores de Decisão e as Florestas Aleatórias.

Conceitos Fundamentais

- **Nó Raiz (Root Node)**: O nó superior da árvore, que não possui pai. É o ponto de partida para a travessia da árvore.
- **Nó Filho (Child Node)**: Um nó diretamente conectado a outro nó, que é seu pai.
- **Nó Pai (Parent Node)**: Um nó que tem um ou mais nós filhos.
- **Nó Folha (Leaf Node)**: Um nó que não possui filhos. Em árvores de decisão, os nós folha representam as decisões finais ou as classes previstas.
- **Subárvore (Subtree)**: Uma seção da árvore que consiste em um nó pai e todos os seus descendentes.
- **Profundidade (Depth)**: O número de arestas do nó raiz até um nó específico.
- **Altura (Height)**: O número de arestas do nó mais longo (do nó raiz até o nó folha mais distante).

Tipos de Árvores Binárias

Existem vários tipos de árvores binárias, cada uma com propriedades específicas:

- **Árvore Binária Completa**: Todos os níveis da árvore, exceto possivelmente o último, estão completamente preenchidos, e todos os nós no último nível estão o mais à esquerda possível.
- **Árvore Binária Cheia**: Cada nó tem zero ou dois filhos.
- **Árvore Binária Perfeita**: Uma árvore binária cheia onde todos os nós folha estão na mesma profundidade.
- **Árvore Binária Balanceada**: A altura das subárvores esquerda e direita de qualquer nó difere em no máximo um.

Aplicações em Mineração de Dados

As árvores binárias são amplamente utilizadas em algoritmos de classificação e regressão. As Árvores de Decisão, por exemplo, constroem uma estrutura de árvore binária onde cada nó interno representa um teste em um atributo, cada ramo representa o resultado do teste e cada nó folha representa uma classe ou valor de previsão. A simplicidade e interpretabilidade das árvores binárias as tornam ferramentas poderosas para a descoberta de padrões e a tomada de decisões baseada em dados.

Classificação em Mineração de Dados

A classificação é uma das tarefas mais comuns e importantes na mineração de dados, focada em prever a qual categoria ou classe um novo ponto de dados pertence, com base em um conjunto de dados de treinamento com classes conhecidas. É um tipo de aprendizado supervisionado, onde o algoritmo aprende a mapear atributos de entrada para uma classe de saída.

Conceitos Fundamentais

- **Variável Alvo (Target Variable)**: A variável categórica que se deseja prever. Por exemplo, "comprador" (sim/não), "tipo de doença" (A, B, C), "aprovação de crédito" (aprovado/reprovado).

- **Variáveis Preditivas (Predictor Variables)**: Os atributos de entrada usados para prever a variável alvo.
- **Modelo de Classificação**: O algoritmo ou função que aprende a relação entre as variáveis preditivas e a variável alvo.
- **Conjunto de Treinamento (Training Set)**: Dados usados para treinar o modelo, onde as classes são conhecidas.
- **Conjunto de Teste (Test Set)**: Dados usados para avaliar o desempenho do modelo em dados não vistos.

Etapas do Processo de Classificação

1. **Preparação dos Dados**: Inclui limpeza, pré-processamento e transformação dos dados, como já discutido, para garantir que estejam em um formato adequado para o algoritmo de classificação.
2. **Divisão dos Dados**: O conjunto de dados é dividido em conjuntos de treinamento e teste. Uma divisão comum é 70-30% ou 80-20% para treinamento e teste, respectivamente.
3. **Seleção do Algoritmo**: Escolha do algoritmo de classificação mais apropriado para o problema (e.g., Árvores de Decisão, SVM, Naive Bayes, Redes Neurais).
4. **Treinamento do Modelo**: O algoritmo é alimentado com o conjunto de treinamento para aprender os padrões e construir o modelo.
5. **Avaliação do Modelo**: O modelo treinado é testado no conjunto de teste para medir seu desempenho usando métricas apropriadas.
6. **Implantação (Opcional)**: Se o desempenho for satisfatório, o modelo pode ser implantado para fazer previsões em novos dados.

Métricas de Avaliação de Modelos de Classificação

A avaliação do desempenho de um modelo de classificação é crucial. As métricas mais comuns são:

- **Acurácia (Accuracy)**: Proporção de previsões corretas sobre o total de previsões. É uma métrica simples, mas pode ser enganosa em conjuntos de dados desbalanceados. `Acurácia = (Verdadeiros Positivos + Verdadeiros Negativos) / Total de Observações`

- **Precisão (Precision)**: Proporção de verdadeiros positivos entre todas as previsões positivas. Responde à pergunta: "Das vezes que o modelo previu positivo, quantas estavam corretas?" `Precisão = Verdadeiros Positivos / (Verdadeiros Positivos + Falsos Positivos)`
- **Recall (Sensibilidade)**: Proporção de verdadeiros positivos entre todas as observações positivas reais. Responde à pergunta: "Das observações que eram realmente positivas, quantas o modelo identificou corretamente?" `Recall = Verdadeiros Positivos / (Verdadeiros Positivos + Falsos Negativos)`
- **F1-Score**: Média harmônica da precisão e do recall. É útil quando se busca um equilíbrio entre precisão e recall, especialmente em classes desbalanceadas. `F1-Score = 2 * (Precisão * Recall) / (Precisão + Recall)`
- **Matriz de Confusão (Confusion Matrix)**: Uma tabela que resume o desempenho de um algoritmo de classificação. Ela mostra o número de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos.
- **Curva ROC e AUC (Receiver Operating Characteristic and Area Under the Curve)**: A curva ROC plota a taxa de verdadeiros positivos (Recall) versus a taxa de falsos positivos em vários limiares de classificação. A AUC mede a área sob a curva ROC, fornecendo uma medida agregada do desempenho do classificador em todos os limiares. Um valor de AUC próximo de 1 indica um excelente classificador.

Árvores de Decisão como Classificadores

As árvores de decisão são algoritmos de classificação populares devido à sua interpretabilidade. Elas funcionam dividindo o conjunto de dados em subconjuntos menores com base em testes de atributos, formando uma estrutura de árvore. Cada nó interno representa um teste em um atributo, cada ramo representa o resultado do teste e cada nó folha representa uma decisão de classe.

- **Parâmetros Importantes**: Profundidade máxima da árvore (`max_depth`), número mínimo de amostras para dividir um nó (`min_samples_split`), número mínimo de amostras em uma folha (`min_samples_leaf`).
- **Poda (Pruning)**: Técnica para reduzir o tamanho da árvore de decisão, removendo ramos que fornecem pouco poder preditivo e podem levar ao overfitting. A poda pode ser pré-poda (parar a construção da árvore antes que

ela esteja totalmente crescida) ou pós-poda (construir a árvore completa e depois remover ramos).