Apostila Teórica: Fundamentos de Machine Learning e Pré-processamento de Dados

Esta apostila possui caráter teórico. Seu objetivo é ajudar você a entender **por que** realizamos determinadas etapas no pré-processamento de dados e em Machine Learning, bem como fornecer referências para que você possa aprofundar seus estudos. Ao final de cada tópico, apresentamos uma sugestão de ferramenta (classe, método ou função) de uma biblioteca para que você possa pesquisar detalhadamente a respeito.

Por que Realizamos o Pré-processamento de Dados?

Antes de alimentarmos um modelo de Machine Learning com dados, é essencial que esses dados estejam em um formato compreensível e adequado. Diferentes modelos têm exigências específicas, e a qualidade dos dados é determinante no desempenho final. Dados "crus" geralmente contêm problemas como valores ausentes, formatos não padronizados e distribuições muito diferentes entre atributos. O pré-processamento consiste em uma série de etapas que visam resolver ou atenuar esses problemas, permitindo que o modelo se concentre no aprendizado de padrões úteis ao invés de lidar com "sujeira" ou inconsistências nos dados.

Tratamento de Valores Ausentes

Por que remover ou imputar valores ausentes?

Valores ausentes representam informações que não foram registradas ou disponíveis durante a coleta de dados. Podem surgir por diversos motivos: falhas técnicas, falta de resposta de um entrevistado, erro humano, entre outros. Esses "buracos" podem causar problemas durante o treinamento do modelo, pois muitos algoritmos não sabem lidar diretamente com valores ausentes. Além disso, a presença de dados incompletos pode introduzir viés ou levar o modelo a interpretar incorretamente as relações entre variáveis.

Existem duas abordagens comuns:

1. **Remoção de Valores Ausentes**:
   Remover linhas ou colunas inteiras que contenham valores ausentes.
   Vantagem: É simples e direta.
   Desvantagem: Se muitas informações forem removidas, isso pode diminuir a representatividade dos dados ou até mesmo eliminar padrões importantes.

2. **Imputação de Valores**:
   Substituir valores ausentes por uma estimativa. Essa estimativa pode ser uma

média, mediana, moda, ou até mesmo ser resultado de um modelo estatístico ou de Machine Learning.
Vantagem: Mantém a maior quantidade de dados possível.
Desvantagem: Ao "adivinhar" valores, corre-se o risco de introduzir viés ou mascarar a variabilidade verdadeira do conjunto de dados.

Exemplo de Ferramenta

- **Pandas:** `DataFrame.dropna()`
  Documentação:
  https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html

  Essa função remove valores ausentes do DataFrame, seja por linha ou por coluna.

Codificação de Variáveis Categóricas

Por que transformar variáveis categóricas em numéricas?

Muitos algoritmos de Machine Learning trabalham internamente apenas com valores numéricos. Variáveis categóricas (como "Cor", podendo ser "Vermelho", "Azul", "Verde") precisam ser convertidas para uma representação numérica, sem perder a informação original, antes de serem utilizadas pelo modelo.

A codificação tem a função de garantir que cada categoria seja entendida de forma adequada. Um dos métodos mais comuns é o **One-Hot Encoding**, onde cada categoria se transforma em uma coluna binária (0 ou 1). Isso evita que o modelo interprete categorias ordinais quando elas não são (por exemplo, atribuir "Vermelho" = 1 e "Azul" = 2 poderia sugerir que Azul > Vermelho, o que não faz sentido em muitos contextos).

Essa etapa é essencial para que o algoritmo possa "entender" adequadamente a informação categórica. Sem a codificação, o modelo pode falhar em reconhecer padrões fundamentais presentes nos dados não-numéricos.

Exemplo de Ferramenta

- **Pandas:** `get_dummies()`
  Documentação:
  https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html

  Esta função transforma variáveis categóricas em colunas dummy (binárias), realizando o One-Hot Encoding.

Normalização e Padronização de Dados Numéricos

Por que normalizar ou padronizar dados?

Em muitos conjuntos de dados, diferentes atributos podem ter escalas bastante distintas. Por exemplo, a renda (que pode variar em milhares de unidades monetárias) e o número de filhos (geralmente um dígito pequeno) estão em magnitudes completamente diferentes. Se um modelo não considerar essa disparidade, os atributos com valores mais altos podem dominar o processo de aprendizagem, ofuscando variáveis igualmente importantes, porém com valores numéricos menores.

A normalização e a padronização têm como objetivo colocar todos os atributos em uma escala comparável:

- **Normalização (Min-Max Scaling)**: Converte os valores para uma escala entre 0 e 1.
  Vantagem: Garante todos os valores em uma faixa padrão simples.
  Desvantagem: Pode ser sensível a outliers, pois valores extremos alteram significativamente a escala.

- **Padronização (Standard Scaling)**: Transforma os dados para que tenham média 0 e desvio padrão 1.
  Vantagem: Reduz a influência de outliers e é utilizada por diversos algoritmos que assumem distribuição normal dos dados.
  Desvantagem: Pode não ser apropriada se a distribuição for muito distorcida.

Essas técnicas são particularmente importantes para algoritmos baseados em distância (como k-NN) ou que utilizam regularização (como regressão logística e SVM), já que os atributos em escalas semelhantes evitam que um atributo domine o cálculo da distância ou do erro.

Exemplo de Ferramenta

- **Scikit-Learn: `MinMaxScaler`**
  Documentação:
  https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

  Este scaler do scikit-learn normaliza cada recurso (coluna) de modo que todos os valores fiquem entre um valor mínimo e máximo (geralmente 0 e 1).

Separação do Conjunto de Dados em Treino e Teste

Por que dividir o dataset em treino e teste?

Um dos objetivos principais do aprendizado de máquina é a capacidade de **generalização**: o modelo não deve apenas memorizar os dados nos quais foi treinado, mas também

performar bem em dados nunca antes vistos. Ao separar o conjunto de dados em duas (ou mais) partes, podemos treinar o modelo em um subconjunto (treino) e, posteriormente, avaliar seu desempenho em outro subconjunto (teste), que não foi utilizado durante o treinamento.

Dessa forma, é possível identificar se o modelo realmente aprendeu padrões úteis ou se simplesmente decorou exemplos específicos (overfitting). A validação em um conjunto de teste fornece uma visão mais honesta da capacidade de generalização do modelo.

Exemplo de Ferramenta

- **Scikit-Learn:** `train_test_split()`
  Documentação:
  https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

  Este método divide o dataset em subsets de treino e teste de forma aleatória, garantindo também a opção de estratificação para manter a proporção das classes.

Conclusão

O pré-processamento de dados não é apenas um passo burocrático, mas sim um procedimento fundamental que impacta diretamente na qualidade do modelo final. Investir tempo na compreensão de **por que** cada etapa é necessária ajuda a tomar decisões informadas ao preparar dados e garante maior confiança nos resultados obtidos pelo modelo de Machine Learning.

Lembre-se de consultar a documentação oficial das bibliotecas e ferramentas sugeridas para entender as nuances, parâmetros e opções disponíveis, garantindo assim que você utilize a abordagem mais adequada ao seu problema.