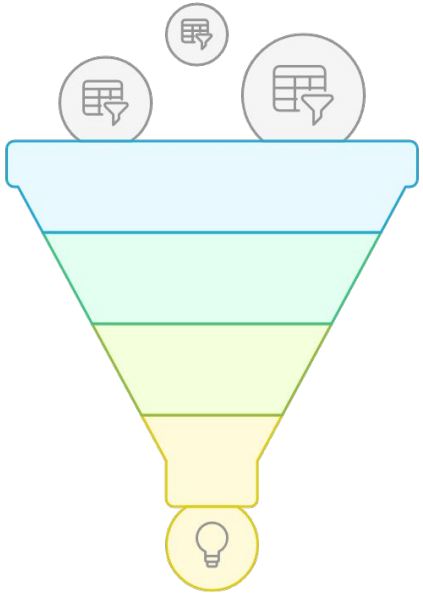
KDD

Flavio Motta
flavioaam@hotmail.com

O que é KDD - Knowledge-Discovery in Databases

- Processo de Descoberta de Conhecimento em Bancos de Dados
  - Engloba técnicas de ciência da computação, estatística e inteligência artificial para extrair conhecimento útil e compreensível.



Pré-processamento de Dados

Limpando e organizando dados para análise

Análise Estatística

Aplicando métodos estatísticos para insights

Aprendizado de Máquina

Usando algoritmos para modelagem de dados

Interpretação de Resultados

Traduzindo dados analisados em insights acionáveis

Origem e Necessidade

- Na era digital, a quantidade de dados gerada por diversas fontes é imensa.
- A necessidade de transformar esses dados brutos em informação compreensível e acionável levou ao desenvolvimento do KDD.
- Permite tomar decisões informadas, identificar padrões ocultos, prever tendências futuras e otimizar processos.

Etapas do processo KDD

O KDD é um processo que inclui várias etapas, desde a seleção de dados até a avaliação dos padrões descobertos.

1. **Seleção de Dados**: Identificar e coletar dados relevantes.
2. **Limpeza e o pré-processamento de dados**: Realização da limpeza dos dados e estratégias para lidar com campos de dados ausentes.
3. **Transformação de dados**: Etapa para encontrar formas de representar os dados com base no objetivo da análise.
    - Realizando redução no número de variáveis.

Etapas do processo KDD

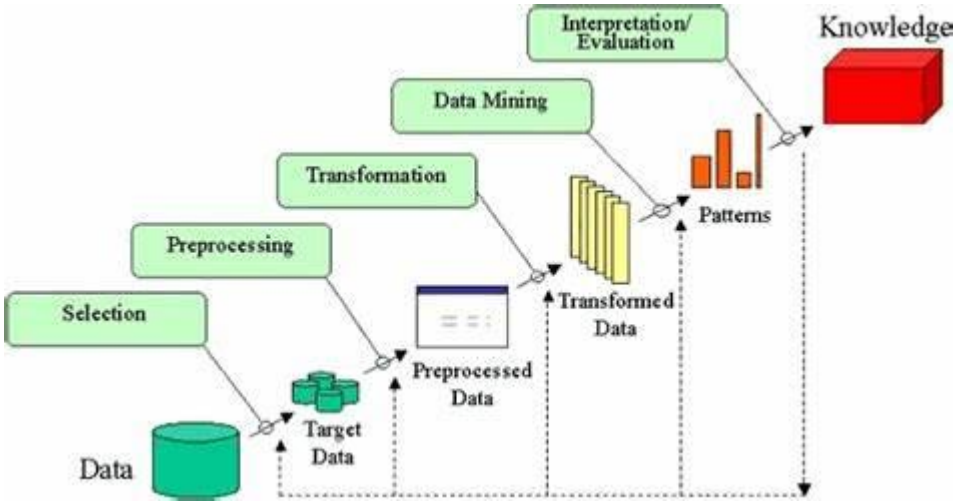
4. **Escolha do método de mineração de dados**: Escolher o método de mineração de dados com base nos objetivos da aplicação (Por exemplo: sumarização, classificação, regressão, agrupamento, etc.)
5. **Análise exploratória e a seleção de modelos e hipóteses**: Escolher o(s) algoritmo(s) de datamining e selecionar o(s) método(s) a serem usados para a busca de padrões nos dados.
6. **Mineração de dados**: Busca por padrões de interesse em uma forma representacional específica ou um conjunto de tais representações, incluindo regras de classificação, regressão e agrupamento.

Etapas do processo KDD

7. **Interpretar padrões extraídos**: Esta etapa envolve interpretar os padrões e modelos extraídos.
8. **Agir com base no conhecimento descoberto**: Usar diretamente as informações descobertas e/ou incorporar o conhecimento em outro sistema para ação posterior ou simplesmente documentá-lo e relatá-lo às partes interessadas.

Mineração de Dados x KDD

Enquanto a mineração de dados se refere especificamente à etapa de análise e extração de padrões dentro do processo de KDD, KDD é o processo completo, que inclui não apenas a mineração, mas também a preparação e interpretação dos dados.

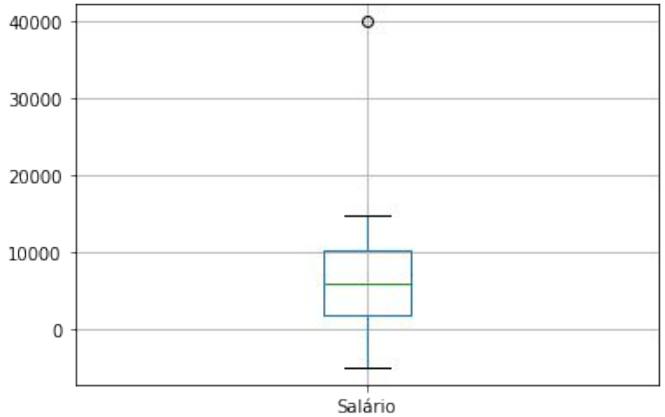


Limpeza, preparação e exploração de dados

Objetivo

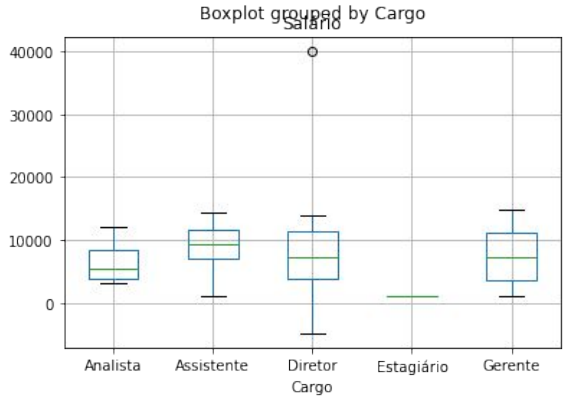
- Dados limpos são cruciais para modelos precisos e insights confiáveis.
- Desafios Comuns: Valores ausentes, duplicados, erros de entrada e outliers.

	Nome	Nascimento	Salário	Cargo
99	JÚLIO OLIVEIRA	25/01/1990	11959.0	Analista
100	JÚLIO OLIVEIRA	25/01/1990	11959.0	Analista
101	NaN	05/08/1990	1000.0	Estagiário
102	JÚLIO MASSI	NaN	1000.0	Estagiário
103	MAYCON CRUZ	21/04/1999	NaN	Estagiário
104	FILIPE CRUZ	01/08/2001	NaN	Gerente
105	PEDRO NÁPOLES	05/13/1990	1000.0	Gerente
106	OTÁVIO SOARES	05/11/1790	1000.0	Asistente
107	VICTOR RODRIGUES	05/11/2990	40000.0	Direto
108	VANESSA OLIVEIRA	105/11/1990	-5000.0	Diretor



Técnicas de limpeza

- Tratamento de Valores Ausentes:
  - Métodos como imputação (média, mediana, mais frequente) e exclusão de registros.




```
# Descobrindo valores ausentes
df.isnull().sum()
# Imprimindo as linhas com valores ausentes
print(df[df.isnull().any(axis=1)])
# Média de salários por cargo
print(df.groupby('Cargo').mean())
# Mediana de salários por cargo
print(df.groupby('Cargo').median())
# Boxplot de salários por cargo
df.boxplot(column='Salário', by='Cargo')
plt.show()
# Preenchendo valores ausentes com a média dos salários dos cargos
df['Salário'] =
df['Salário'].fillna(df.groupby('Cargo')['Salário'].transform('mean'))
# Imprimindo as linhas com valores ausentes
print(df[102:105])
```

Técnicas de limpeza

- Correção de Duplicatas:
  - Como identificar e corrigir registros duplicados.

```
# Imprimindo quantidades de duplicados
print(df.duplicated().sum())
# Identificando duplicados
print(df[df.duplicated()])
# Imprimindo as linhas duplicadas
print(df[df.duplicated(keep=False)])
# Removendo duplicados
df = df.drop_duplicates()
```

Técnicas de limpeza - Correção de Erros de Entrada

```
#Identificando erros de digitação
print(df['Cargo'].unique())
```

```
['Assistente' 'Estagiário' 'Gerente' 'Diretor' 'Analista' 'Asistente'
 'Direto']
```

```
df['Cargo'] = df['Cargo'].str.replace('Direto','Diretor')
df['Cargo'] = df['Cargo'].str.replace('Asistente','Assistente')
print(df['Cargo'].unique())
```

```
['Assistente' 'Estagiário' 'Gerente' 'Diretorr' 'Analista' 'Diretor']
```

```
df['Cargo'] = df['Cargo'].str.replace(r'\bDireto\b','Diretor')
```

Técnicas de limpeza - Correção de Erros de Entrada

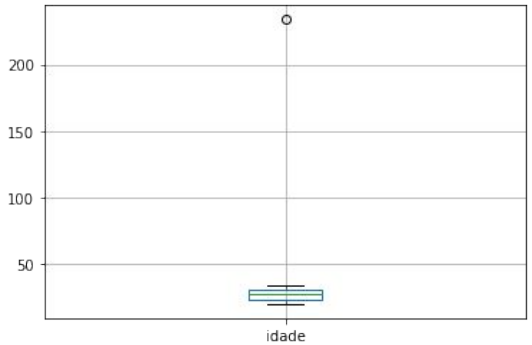
```
df_datas['data'] = pd.to_datetime(df_datas['data'],
errors='coerce')
# Identificar as linhas onde a conversão resultou em NaT
linhas_com_erro = df_datas[df_datas['data'].isna()]

if not linhas_com_erro.empty:
    print("Linhas com datas não convertíveis para
datetime:")
    print(linhas_com_erro)

df_datas.dropna(subset=['data'], inplace=True)
df_datas['dia'] = df_datas['data'].dt.day
df_datas['mes'] = df_datas['data'].dt.month
df_datas['ano'] = df_datas['data'].dt.year
df_datas['idade'] = 2024 - df_datas['ano']
```

Técnicas de limpeza - Correção de Erros de Entrada

```
#imprimir a média de idade por cargo
print(df_datas.groupby('cargo')['idade'].mean())
cargo
Analista      25.642857
Assistente    36.571429
Diretor       28.150000
Estagiário    27.350000
Gerente       27.400000
df_datas.boxplot(column='idade')
```




```
print(df_datas[df_datas['idade'] > 100])
```

Técnicas de limpeza

```
# 102,106,107,108 São linhas com erros de data, podemos preencher
com a mediana por cargo
media_diretor = 29
media_assistente = 27
media_estagiario = 28
nascimento_diretor = '01/01/1995'
nascimento_assistente = '01/01/1997'
nascimento_estagiario = '01/01/1996'
df.loc[102, 'Nascimento'] = nascimento_estagiario
df.loc[106, 'Nascimento'] = nascimento_assistente
df.loc[107, 'Nascimento'] = nascimento_diretor
df.loc[108, 'Nascimento'] = nascimento_diretor

df[101:108]
```

Técnicas de limpeza - Normalização e Padronização

- Normalização e padronização são fundamentais no pré-processamento de dados, especialmente em contextos de machine learning.
  - Essas técnicas ajudam a transformar os dados para que estejam em uma escala comum, o que é crucial para algoritmos que são sensíveis à escala das variáveis ou que assumem que os dados estão distribuídos de maneira específica.

Técnicas de limpeza - Normalização de Variáveis Contínuas

- Por que é necessário:
  - **Sensibilidade à Escala**: Muitos algoritmos de machine learning, como K-Nearest Neighbors (KNN), gradient descent, redes neurais e outros baseados em distância ou gradientes, são sensíveis às escalas das variáveis. Se uma variável varia de 0 a 1000 enquanto outra varia de 0 a 1, o algoritmo pode indevidamente dar mais importância à variável com maior escala.
  - **Convergência Mais Rápida**: Em algoritmos que usam otimização, como gradient descent, variáveis em escalas comuns podem ajudar o algoritmo a convergir mais rapidamente.

Técnicas de limpeza - Padronização de Variáveis Categóricas

- A padronização de variáveis categóricas envolve transformá-las em um formato que pode ser fornecido a algoritmos ML, geralmente convertendo-as em números.
- Por que é necessário:
  - **Compatibilidade com Algoritmos**: Muitos algoritmos de machine learning esperam entrada numérica e não podem trabalhar diretamente com categorias rotuladas como strings.
  - **Relações entre Categorias**: Transformar categorias em variáveis numéricas ou indicadoras permite que o modelo capture informações sobre a presença ou ausência de uma categoria, além de possibilitar o uso de relações ordinais quando apropriado.

Técnicas de limpeza - Padronização de Variáveis Categóricas

- Técnicas Comuns:
  - **Label Encoding**: Cada categoria é atribuída a um número inteiro.
    - Útil para variáveis categóricas ordinais.
  - **One-Hot Encoding**: Cria uma nova coluna para cada categoria, com 1 onde a categoria está presente e 0 onde não está.
    - Útil para variáveis nominais sem ordem inerente.