

Sdco



# PRÉ PROCESSAMENTO DE DADOS

# PRÉ-PROCESSAMENTO



OS MÉTODOS DE COLETA DE DADOS  
GERALMENTE SÃO POUCO CONTROLADOS



# PRÉ-PROCESSAMENTO



DADOS IMPOSSÍVEIS (SEXO: MASCULINO, GRÁVIDA: SIM);

DADOS AUSENTES;

DADOS EM FORMATO INCORRETO;

DADOS IRRELEVANTES;

DADOS REDUNDANTES;

DADOS RUIDOSOS E NÃO CONFIÁVEIS;

ETC.



# PRÉ-PROCESSAMENTO



MUITAS VEZES, O PRÉ-PROCESSAMENTO DE DADOS É A **FASE MAIS IMPORTANTE** DE UM PROJETO DE APRENDIZADO DE MÁQUINA. ESTUDOS PONTAM PARA 70 A 80% DO ESFORÇO EM UM PROJETO DE CIÊNCIA DE DADOS.



# PRÉ-PROCESSAMENTO



O PRÉ-PROCESSAMENTO DE DADOS INCLUI LIMPEZA E SELEÇÃO DE INSTÂNCIAS, NORMALIZAÇÃO, TRANSFORMAÇÃO, EXTRAÇÃO (DADOS DERIVADOS) E SELEÇÃO DE CARACTERÍSTICAS.



# PRÉ-PROCESSAMENTO



O PRÉ-PROCESSAMENTO É NORMALMENTE  
TRATADO *FEATURE A FEATURE*.  
EM CASOS ESPECIAIS PODEM-SE  
COMBINAR *FEATURES* PARA O  
TRATAMENTO.



# PRÉ-PROCESSAMENTO



O PANDAS É UMA DAS MELHORES  
FERRAMENTAS PARA ABERTURA E PRÉ-  
PROCESSAMENTO DE DADOS.

```
import pandas as pd
```





COMO

Sdco

# LIMPAR?

MEUS  
DADOS



# MISSING VALUES



# MISSING VALUES



VALORES AUSENTES OCORREM  
QUANDO NENHUM VALOR DE DADOS  
É ARMAZENADO PARA A VARIÁVEL  
EM UMA OBSERVAÇÃO.



# TÉCNICAS MISSING VALUES



1 – EXCLUSÃO DA OBSERVAÇÃO

2 – IMPUTAÇÃO DE MÉDIA

3 – IMPUTAÇÃO MEDIANA

4 – INTERPOLAÇÃO

5 – IMPUTAÇÃO POR MACHINE LEARNING



## CONTANDO MISSING VALUES

```
df['sua_feature'].isnull().sum()
```

### **Multiples features**

```
df[['sua_feature','outra_feature']].isnull().sum()
```

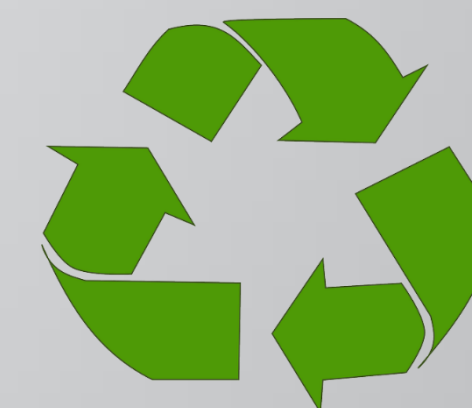


TRATANDO



**MISSING**

VALUES



# EXCLUSÃO TOTAL

# EXCLUSÃO TOTAL



UMA OPÇÃO É EXCLUIR TODAS AS  
'FEATURES' COM VALORES AUSENTES

```
df.dropna(inplace=True)
```

Ou

```
df = df.dropna()
```





# EXCLUIR MISSING VALUES



ENTRETANTO, PODE-SE PERDER MUITAS  
OBSERVAÇÕES PREJUDICANDO O  
MODELO. VAMO AVALIAR A PERDA.

```
df.describe()
```

```
df.dropna().describe()
```

Execute em células separadas



# INPUTANDO MÉDIA



# IMPUTANDO MÉDIA



A MÉDIA É MUITO IMPACTADA POR  
OUTLIERS. UMA MÉTRICA MAIS  
ROBUSTA SERIA A MEDIANA



## IMPUTAÇÃO DA MÉDIA

Em somente uma 'feature'

```
df.loc[df['feature'].isnull() , 'feature'] = df['feature'].median()
```

Podemos usar `fillna()` para substituir os valores ausentes pelo valor médio de cada coluna

```
df.fillna(df.media() , inplace=True)
```



# INTERPOLAÇÃO



# INTERPOLAÇÃO



USADA QUANDO EXISTE FALHAS NA  
FREQUÊNCIA DE SERIES TEMPORAIS



# INTERPOLAÇÃO



```
series = read_csv('minha_serie.csv')  
upsampled = series.resample('D')
```

D = Diário

M = Mensal

A = Anual (annual)



# IMPUTAÇÃO POR MODELO





# IMPUTAÇÃO POR MODELO



ALGUM MODELO DE MACHINE  
LEARNING PODE SER USADO PARA  
IMPUTAÇÃO DE DADOS.



# IMPUTAÇÃO POR MODELO



OS MAIS COMUNS SÃO:

- TREE DECISION
- NAÏVE BAYES
- LINEAR REGRESSION



# IMPUTAÇÃO POR MODELO



NESSE CASO SEPARA-SE UMA OU  
MAIS COLUNAS QUE CONTENHA BOA  
CORRELAÇÃO COM A DOS DADOS  
FALTANTES E QUE ESTEJA  
COMPLETA. SEM DADOS FALTANTES.



# IMPUTAÇÃO POR MODELO



NESSE CASO SEPARA-SE UMA OU  
MAIS COLUNAS QUE CONTENHA BOA  
CORRELAÇÃO COM A DOS DADOS  
FALTANTES E QUE ESTEJA  
COMPLETA. SEM DADOS FALTANTES.



# IMPUTAÇÃO POR MODELO



A BIBLIOTECA SCIKIT-LEARN FORNECE A  
CLASSE DE PRÉ-PROCESSAMENTO  
**IMPUTER()** QUE PODE SER USADA PARA  
SUBSTITUIR VALORES AUSENTES BASEADO  
EM MODELOS.

A classe Imputer opera diretamente no array NumPy em vez do DataFrame.



# IMPUTAÇÃO POR MODELO



```
from sklearn.preprocessing import Imputer  
imputer = Imputer()
```

```
X_imputed = imputer.fit_transform(X)
```



# IMPUTAÇÃO MULTIPLAS



Técnica desenvolvida por Rubin (1987) e é uma técnica baseada em regressão linear Bayesiana (BLR- Bayesian Linear Regression), método da média preditiva (PMM-Predictive Mean Matching) para padrão monotônico.

Para padrão não monotônico são: MCMC( Markov Chain Monte Carlo): o método de Monte Carlo baseado em Cadeia de Markov (MCMC)





COMO

Sdco

# TRATAR?

DADOS  
CATEGÓRICOS





# DADOS CATEGÓRICOS



MUITOS ALGORITMOS NÃO PODEM  
LIDAR COM DADOS CATEGÓRICOS



# DADOS CATEGÓRICOS



OS MAIS FAMOSOS:

- REDES NEURAIS
- KNN
- K-MEANS

... ETC



# DADOS CATEGÓRICOS



PORTANTO É IMPORTANTE REALIZAR  
TRANSFORMAÇÕES NOS DADOS  
QUALITATIVOS QUE REPRESENTAM  
CATEGORIAS.



# DADOS CATEGÓRICOS



A TÉCNICAS SÃO:

- BINARIZAÇÃO
- DISCRETIZAÇÃO
- VARIÁVEIS DUMMY



# BINARIZAÇÃO



# BINARIZAÇÃO



É A TRANSFORMAÇÃO DE DADOS  
CATEGORICOS EM VALORES  
BINÁRIOS [0,1]



## BINARIZANDO SEXO - PANDAS

```
df.loc[df['sexo'] == 'MASCULINO', 'sexo'] = 0  
df.loc[df['sexo'] == 'FEMININO', 'sexo'] = 0
```



## BINARIZANDO SEXO – SCIKIT-LEARN

```
from sklearn.preprocessing import LabelEncoder  
encoder = LabelEncoder()
```

```
Sexo = 2 ## sexo é a segunda coluna no array numpy
```

```
X[:,sexo] = encoder.fit_transform(X[:,sexo])
```





# DICRETIZAÇÃO



# DICRETIZAÇÃO



É A TRANSFORMAÇÃO DE DADOS  
CATEGORICOS MULTIVALORADOS  
VALORES DISCRETOS[0,1,2,3...]



# DICRETIZAÇÃO



É A TRANSFORMAÇÃO DE DADOS  
CATEGORICOS MULTIVALORADOS  
VALORES DISCRETOS[0,1,2,3...]



## DISCRETIZANDO – PANDAS

```
df.loc[df['qualidade_ensino'] == 'RUIM', 'qualidade_ensino'] = 0
df.loc[df['qualidade_ensino'] == 'REGULAR', 'qualidade_ensino'] = 1
df.loc[df['qualidade_ensino'] == 'BOA', 'qualidade_ensino'] = 2
df.loc[df['qualidade_ensino'] == 'ÓTIMA', 'qualidade_ensino'] = 3
```



E SE

Sdco

**EXISTIR** ?  
MUITAS  
CATEGÓRIAS



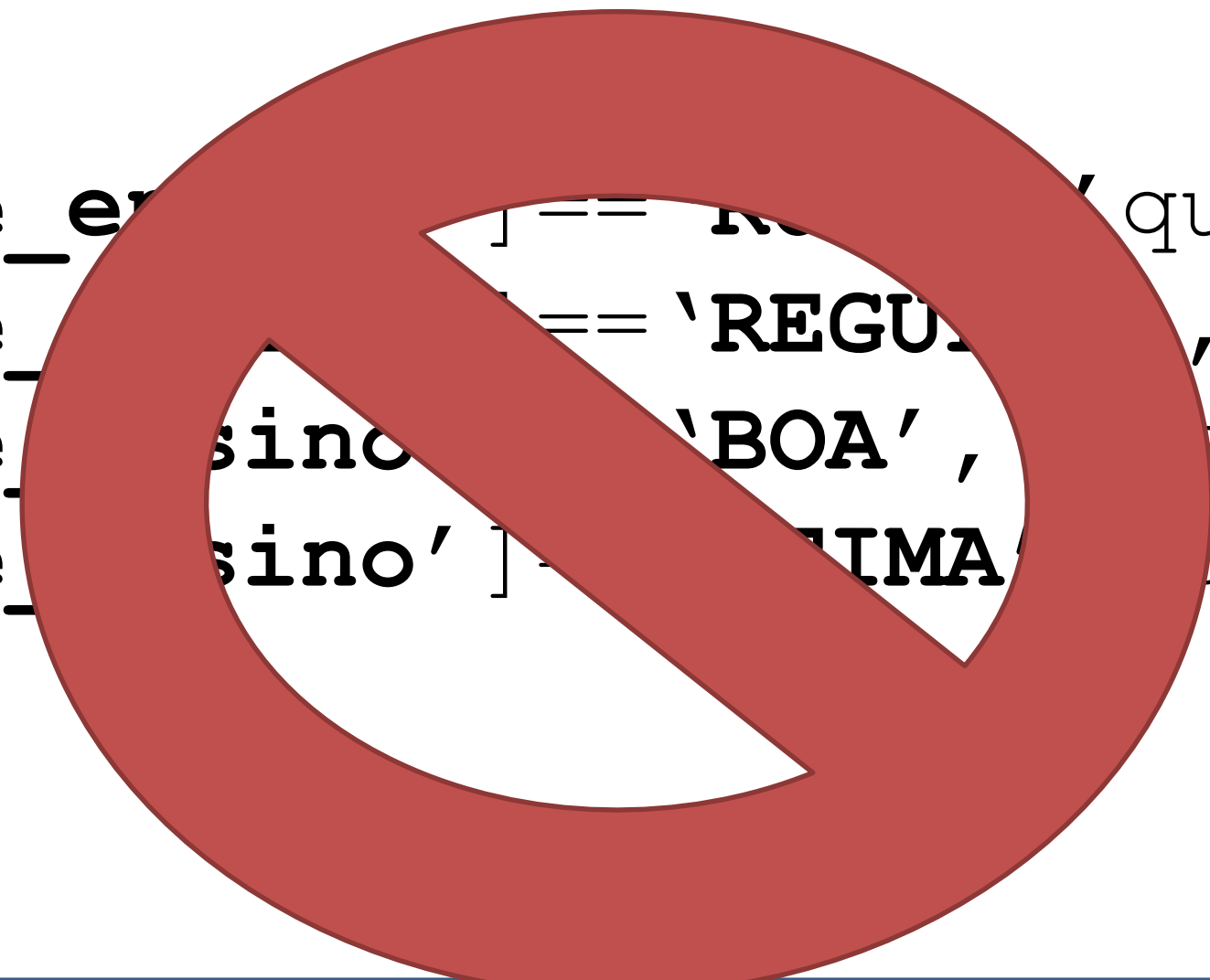
# DICRETIZAÇÃO



ESSA FORMA MANUAL FICA  
IMPRATICÁVEL

## DISCRETIZANDO – PANDAS

```
df.loc[df['qualidade_ensino'] == 'RUIM', 'qualidade_ensino'] = 0  
df.loc[df['qualidade_ensino'] == 'REGULAR', 'qualidade_ensino'] = 1  
df.loc[df['qualidade_ensino'] == 'BOA', 'qualidade_ensino'] = 2  
df.loc[df['qualidade_ensino'] == 'MUITA BOA', 'qualidade_ensino'] = 3
```



# MANIPULANDO CATEGÓRIAS DE UM JEITO MELHOR



# VARIÁVEIS DUMMY





# VARIÁVEIS DUMMY



Indicam a ausência ou a presença de algum efeito categórico que pode ser esperado. Uma variável dummy pode ser considerada como um valor de verdade representado como um valor numérico 0 ou 1.



# VARIÁVEIS DUMMY

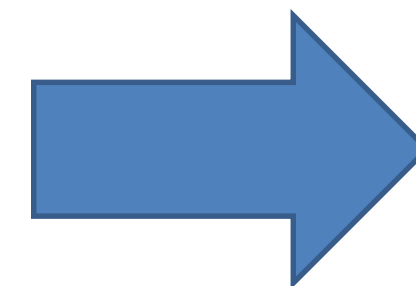


Indicam a ausência ou a presença de algum efeito categórico que pode ser esperado. Uma variável dummy pode ser considerada como um valor de verdade representado como um valor numérico 0 ou 1.



## EXEMPLO: (PANDAS)

OPÇÃO
SIM
NÃO
SIM
TALVEZ
SIM
NÃO
TALVEZ



SIM	NÃO	TALVEZ
1	0	0
0	1	0
1	0	0
0	0	1
1	0	0
0	1	0
0	0	1

```
dummy = pd.get_dummies(df.opção)
```

# VARIÁVEIS DUMMY



APÓS CRIAR AS DUMMIES

- 1 – INCLUIR AS DUMMIES NO DATAFRAME ORIGINAL
- 2 – EXCLUIR CAMPO ORIGINAL
- 3 – TRATAR A “**DUMMY TRAP VARIABLES**”



**CUIDADO**



**COM A**  
**DUMMY**  
**TRAP**

# DUMMY TRAP

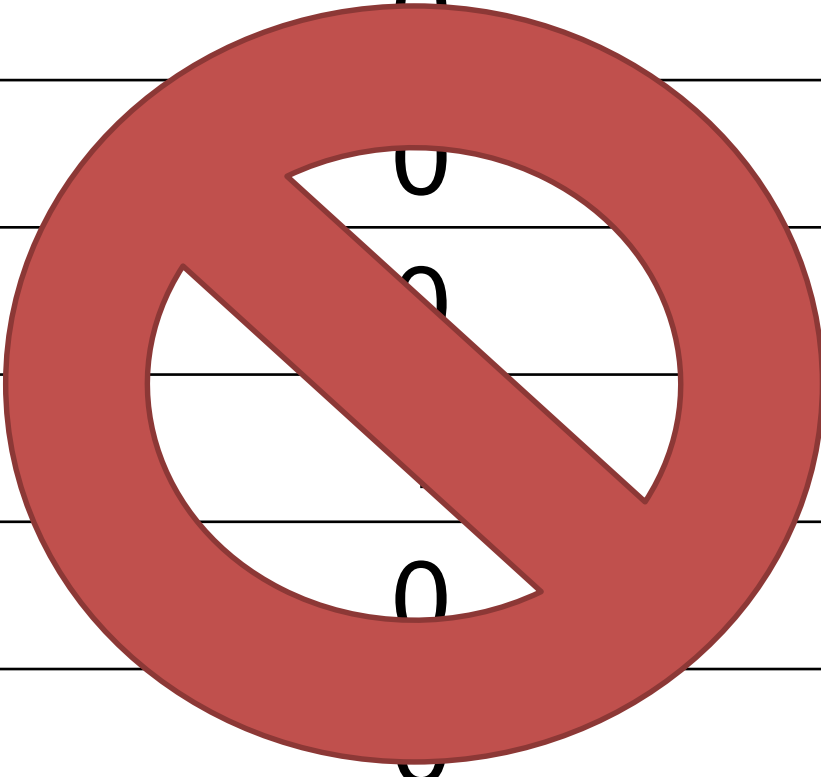


EXISTE UMA SUTIL ARMADILHA QUANDO USAMOS VARIÁVEIS DUMMY. É A REPETIÇÃO DE UM DETERMINADO PADRÃO QUANDO CRIAMOS O NÚMERO DE COLUNAS O MESMO DAS CATEGORIAS.



## EXEMPLO

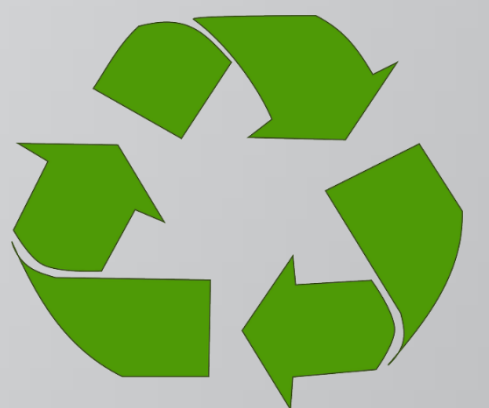
OPÇÃO		SIM	NÃO	TALVEZ
SIM		1	0	0
NÃO		0	1	0
SIM		1	0	0
TALVEZ	TALVEZ (SIM=0, NÃO=0)	0	0	0
SIM		1	0	0
NÃO		0	1	0
TALVEZ	TALVEZ (SIM=0, NÃO=0)	0	0	1



# DUMMY TRAP



PARA RESOLVER O PROBLEMA, BASTA  
EXCLUIR 1 COLUNA DAS DUMMIES  
GERADAS.





## EXEMPLO: SCIKIT-LEARN

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

### PRIMEIRO DISCRETIZAR

```
label = LabelEncoder()
```

```
categoria = 2 ## se a categoria é a segunda coluna no numpy array
```

```
X[:,categoria] = label.fit_transform(X[:, categoria])
```

### GERAR DUMMIES

```
onehotencoder = OneHotEncoder(categorical_features = [1])
```

```
X = onehotencoder.fit_transform(X).toarray()
```

### REMOVENDO UMA CATEGORIA DUMMY TRAP

```
X = X[:, 1:]
```

# NORMALIZAÇÃO



# NORMALIZAÇÃO



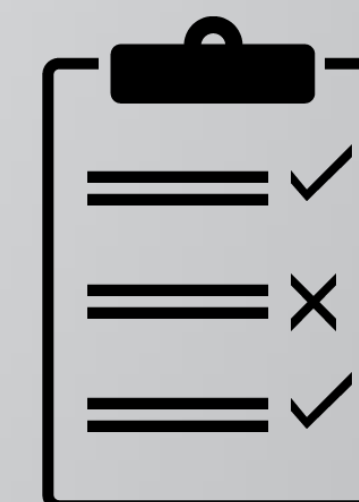
NORMALIZAÇÃO É UM MÉTODO USADO PARA PADRONIZAR O INTERVALO DE VARIÁVEIS INDEPENDENTES OU 'FEATURES' DE DADOS.



# NORMALIZAÇÃO



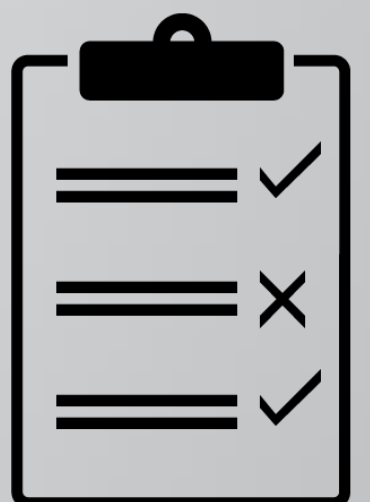
A NORMALIZAÇÃO É UM REESCALONAMENTO DOS DADOS DO INTERVALO ORIGINAL PARA QUE TODOS OS VALORES ESTEJAM DENTRO DO INTERVALO DE 0 E 1.



# NORMALIZAÇÃO



A NORMALIZAÇÃO PODE SER ÚTIL E ATÉ NECESSÁRIA EM ALGUNS ALGORITMOS DE APRENDIZADO DE MÁQUINA QUANDO SEUS DADOS OU SÉRIES TEMPORAIS POSSUEM VALORES DE ENTRADA COM ESCALAS MUITO DISCREPANTES.



# NORMALIZAÇÃO



UMA **FEATURE** COM VALORES MUITO ALTOS EM COMPARAÇÃO A OUTRA **FEATURE** COM VALORES MUITO BAIXOS, PODE TORNAR A PRIMEIRA, COM VALORES ALTOS, UMA **FEATURE** DOMINANTE.

A NORMALIZAÇÃO REDUZ ESSE PROBLEMA.



# NORMALIZAÇÃO



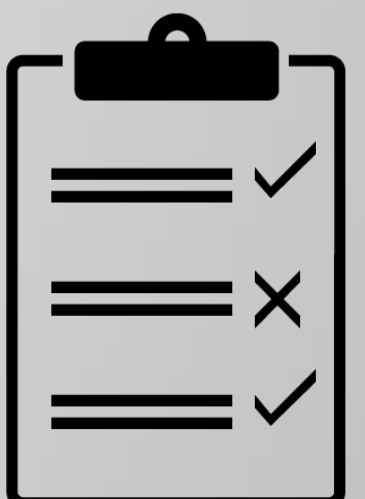
UM VALOR É NORMALIZADO DA SEGUINTE FORMA:

$$X' = (X - \text{MIN}) / (\text{MAX} - \text{MIN})$$

ONDE:

MAX: É O VALOR MÁXIMO DE UMA FEATURE

MIN: É O VALOR MÍNIMO DE UMA FEATURE



## EXEMPLO

```
from sklearn.preprocessing import MinMaxScaler  
  
scaler = MinMaxScaler(feature_range=(0, 1))  
scaler = scaler.fit(X)  
normalized = scaler.transform(X)
```



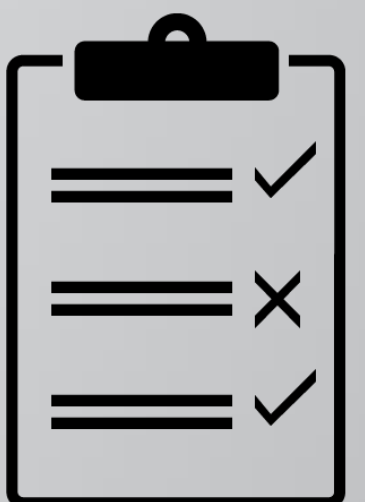
# PADRONIZAÇÃO



# PADRONIZAÇÃO



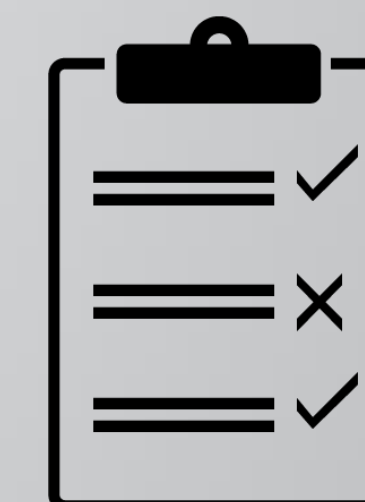
A PADRONIZAÇÃO DE UM CONJUNTO DE DADOS ENVOLVE O REESCALONAMENTO DA DISTRIBUIÇÃO DE VALORES, DE MODO QUE A MÉDIA DOS VALORES OBSERVADOS SEJA 0 E O DESVIO PADRÃO SEJA 1.



# PADRONIZAÇÃO



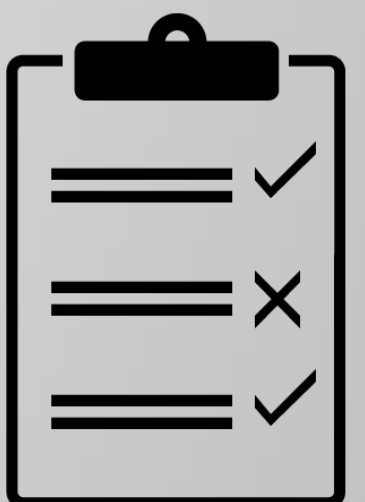
COMO A NORMALIZAÇÃO, A PADRONIZAÇÃO PODE SER ÚTIL E ATÉ NECESSÁRIA EM ALGUNS ALGORITMOS DE APRENDIZADO DE MÁQUINA QUANDO OS DADOS VALORES DE ENTRADA COM ESCALAS DIFERENTES.



# PADRONIZAÇÃO



ALGORITMOS COMO SVM, REGRESSÃO LINEAR  
E LOGÍSTICA, REDES NEURAIS E OUTROS SÃO  
ALGUNS ALGORITMOS QUE MELHORARAM O  
DESEMPENHO COM DADOS GAUSSIANOS.  
MAS NÃO SÃO EM TODOS OS CASOS.  
ESSA AVALIAÇÃO DEVE SER EMPÍRICA.



# PADRONIZAÇÃO



UM VALOR É PADRONIZADO DA SEGUINTE  
FORMA:

$$X' = (X - MÉDIA) / DESVIO\_PADRAO$$

