

# **Tópicos Especiais em Sistemas de Informação**

**Dados de treino x teste,  
avaliação de resultados e variáveis categóricas**

**Ely – [elydasilvamiranda@gmail.com](mailto:elydasilvamiranda@gmail.com)**

# Separando colunas em dataframes

- Após ler os arquivos CSV, é importante separar os dados em subdataframes: dados e marcações;
- Como vimos:
  - As colunas de dados são as que possuem os valores ou perguntas do problema;
  - As marcações são o “alvo”, ou a resposta da combinação das perguntas;
- Há uma nomenclatura que é usar a notação de domínio e imagem da teoria dos conjuntos:
  - $X$  = dados;
  - $Y$  = marcações.

# Estudo de caso: cursos on line

- Um site disponibiliza cursos pagos on line ;
- Gostaríamos de prever se um visitante virá a comprar um curso baseado na sua navegação;
- Basearemos nosso estudo nas seguintes perguntas:
  - Acessou a página principal/home?
  - Acessou a página "como funciona"?
  - Acessou a página de contato?
- A partir dessas perguntas, iremos ter um resultado: comprou ou não comprou um curso.

# Separando colunas em dataframes

- Com os dataframes, é possível separar facilmente esses dois conjuntos;
- Os códigos abaixo copiam todas as linhas das colunas referenciadas:

```
import pandas as pd
```

```
df = pd.read_csv('acessos.csv')
```

```
x = df[['home', 'como_funciona', 'contato']]
```

```
y = df['comprou']
```

```
# ou
```

```
x = df.iloc[:, 0:3]
```

```
y = df.iloc[:, 3]
```

# Separação treino x testes

- Para avaliar resultados, é importante separar os dados em treino e teste;
- Por exemplo, em um CSV, dedicar 75% dos registros pra treino e 25% para testes;
- É importante também conhecermos as marcações/resultados/Y dos testes para aferir o % de acerto;
- Quanto mais registros para treino, melhor será o desempenho do algoritmo.

# Separação treino x testes

- Essa separação pode ser manual, calculando-se o tamanho de cada categoria (treino e testes);
- Ou podemos utilizar uma implementação da biblioteca sklearn:

```
import pandas as pd
df = pd.read_csv('acessos.csv')
x = df[['home', 'como_funciona', 'contato']]
y = df['comprou']
from sklearn.model_selection import train_test_split
x_treino, x_teste, y_treino, y_teste =
    train_test_split(x, y, test_size=0.25, , random_state=0)
```

# Separação treino x testes

- O código anterior retorna 4 subdataframes:
  - x\_treino: as colunas de dados correspondentes a 75% dos registros;
  - x\_teste: as colunas de dados correspondentes a 25% dos registros;
  - y\_treino: a coluna marcações correspondentes a 75% dos registros;
  - y\_teste: a coluna de marcações correspondentes a 25% dos registros;
- O parâmetro random\_state = 0 define que a ordem da separação das colunas não será randômica;

# Avaliação de resultados

- Para avaliar um resultados, existem várias métricas;
- Não necessariamente a melhor métrica é a indicação de % de acerto;
- Abaixo seguem algumas das principais métricas para avaliar resultados de algoritmos de ML:

Precisão	Acurácia
F-measure	Kappa
Recall	Detecção de outliers
Matriz de confusão	
Dentre outras métricas específicas do problema estudado.	



# Avaliação de resultados

- No mínimo, uma avaliação “na mão” deve:
  - Verificar as marcações de testes e compará-las ao que o algoritmo apresentou como resultado;
  - A partir daí, extrair uma porcentagem de acerto;
- Entretanto, isso já está implementado em várias bibliotecas;

# Avaliação de resultados

```
import pandas as pd
df = pd.read_csv('acessos.csv')
x = df[['home', 'como_funciona', 'contato']]
y = df['comprou']
from sklearn.model_selection import train_test_split
x_treino, x_teste, y_treino, y_teste =
    train_test_split(x, y, test_size=0.25, random_state=0)

from sklearn.naive_bayes import MultinomialNB
modelo = MultinomialNB()
modelo.fit(x_treino, y_treino)
modelo.score(x_teste, y_teste)
```

# Variáveis categóricas

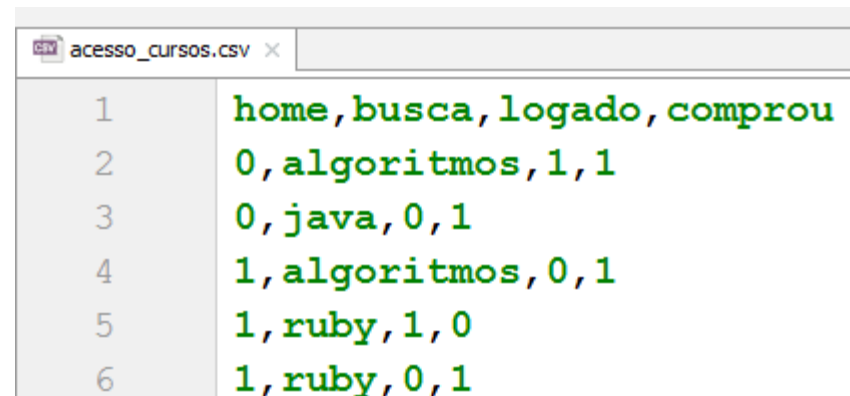
- Para os problemas vistos até agora, sempre usamos variáveis binárias;
- Porém, é muito comum variáveis com mais de duas possibilidades, por exemplo:
  - Grau de instrução: não graduado, graduado, pós-graduado...
  - Renda familiar: entre x e y, entre y e z, entre w e k;
- Tais variáveis não podem ser representadas a princípio com valores binários;
- São chamadas de variáveis categóricas.

# Estudo de caso: busca por um curso específico

- Podemos tornar o nosso modelo um pouco mais real variando uma das perguntas:
  - **home**: se o cliente visitou a página home;
  - **busca**: o que o cliente buscou (algoritmos ou Java ou Ruby ou ...);
  - **estava logado?**: se o cliente estava logado;
  - **comprou?**: se o cliente comprou ou não;
- Perceba que a variável busca agora é categórica por ter mais de duas possibilidades.

# Estudo de caso: busca por um curso específico

- O arquivo `acesso_cursos.csv` possui uma linha de cabeçalho e 1000 linhas de registro;
- Há uma coluna chamada **logado**, indicando se o usuário estava logado ou não
- A coluna **busca** mostra a variável categórica:

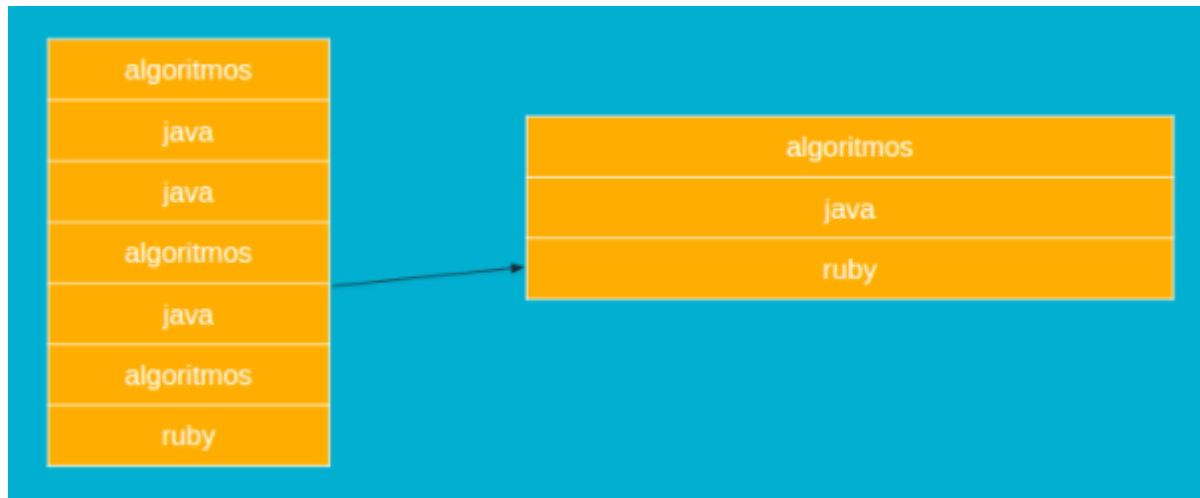


The screenshot shows a CSV file viewer with the title 'acesso\_cursos.csv'. It displays six rows of data. The first row is the header, and the following five rows are data records. Each row is numbered on the left, and the data is shown in a green monospace font.

	home	busca	logado	comprou
1	home	busca	logado	comprou
2	0	algoritmos	1	1
3	0	java	0	1
4	1	algoritmos	0	1
5	1	ruby	1	0
6	1	ruby	0	1

# Variáveis categóricas

- Uma forma de trabalhar com variáveis categóricas é efetuar uma conversão;
- A conversão visa chegar a valores binários;
- O primeiro passo é identificar a quantidade de possíveis categorias de uma variável;
- No nosso estudo de caso, temos 3 cursos disponíveis para a busca:



# Variáveis categóricas

- Neste caso, podemos quebrar a variável busca em 3 variáveis binárias:
  - O cliente buscou algoritmos?
  - O cliente buscou Java?
  - O cliente buscou Ruby?
- Como cada uma tem resposta binária, podemos aplicar normalmente em um algoritmo;
- Assim, teríamos 3 novas "colunas" ou variáveis nos nossos dados.

# Variáveis categóricas

- Para cada possível valor da variável busca:
  - Preenche-se com 1 a ocorrência equivalente;
  - ... e com 0 para o caso das outras ocorrências

home	busca	logado	comprou
1	algoritmos	1	1
1	java	0	0
0	java	0	1
1	ruby	1	1



home	busca	algoritmos	java	ruby	logado	comprou
1	algoritmos	1	0	0	1	1
1	java	0	1	0	0	0
0	java	0	1	0	0	1
1	Ruby	0	0	1	1	1



# Criando variáveis categóricas

- Na biblioteca PANDAS, essas novas colunas de dados são criadas automaticamente;
- Passamos o dataframe dos dados (x) para a função `pandas.get_dummies(x)`;

# Avaliando os resultados

```
import pandas as pd
```

```
df = pd.read_csv('acessos_buscas.csv')
```

```
x = df[['home', 'busca', 'logado']]
```

```
y = df['comprou']
```

```
x = pd.get_dummies(x)
```

```
from sklearn.model_selection import train_test_split
```

```
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size=0.25,  
random_state=0)
```

```
from sklearn.naive_bayes import MultinomialNB
```

```
modelo = MultinomialNB()
```

```
modelo.fit(x_treino, y_treino)
```

```
print(modelo.score(x_teste, y_teste))
```

# **Tópicos Especiais em Sistemas de Informação**

**Dados de treino x teste,  
avaliação de resultados e variáveis categóricas**

**Ely – [elydasilvamiranda@gmail.com](mailto:elydasilvamiranda@gmail.com)**