

# **GANHO DE INFORMAÇÃO E PROCESSAMENTO PARALELO**

COM PYSPARK

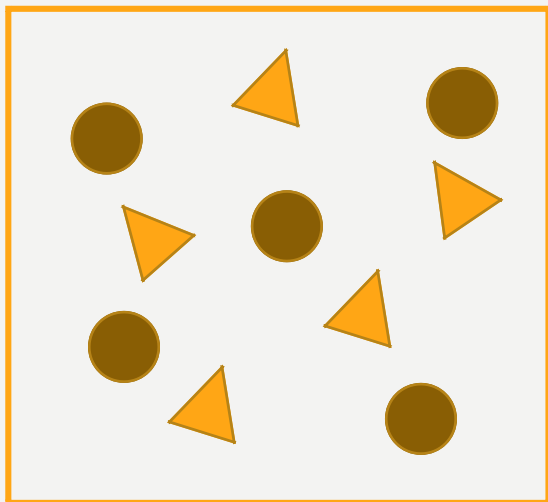


# **GANHO DE INFORMAÇÃO**

# ENTROPIA

- Entropia mede o quão impuro é um conjunto de dados.

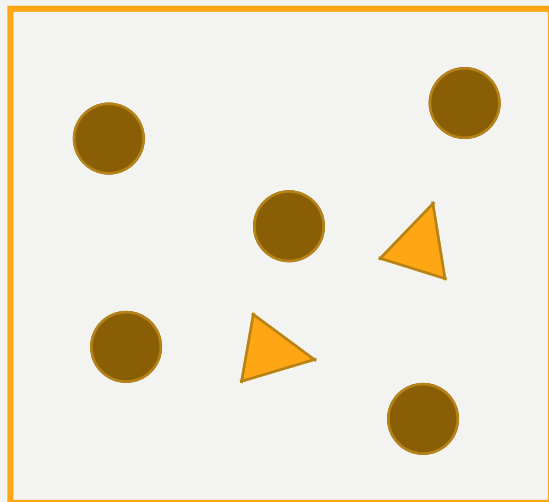
# ENTROPIA



Grupo muito impuro

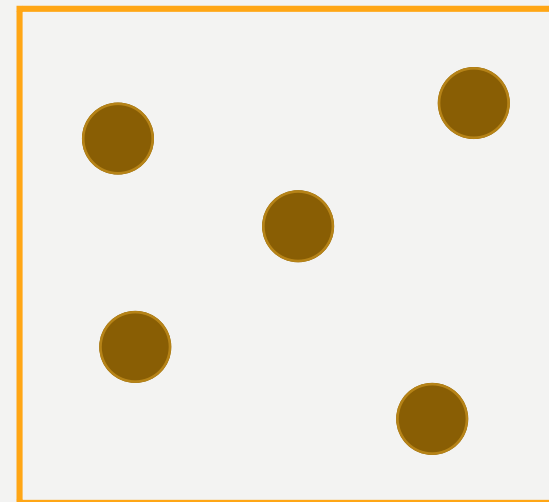
Entropia = 1

Bom grupo para  
aprendizado



Grupo menos impuro

$0 < \text{Entropia} < 1$



Grupo puro

Entropia = 0

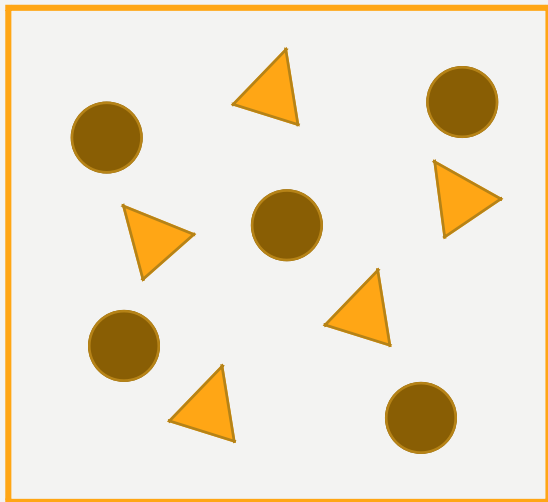
Mau grupo para  
aprendizado

# ENTROPIA

- Entropia de Shannon é definida como

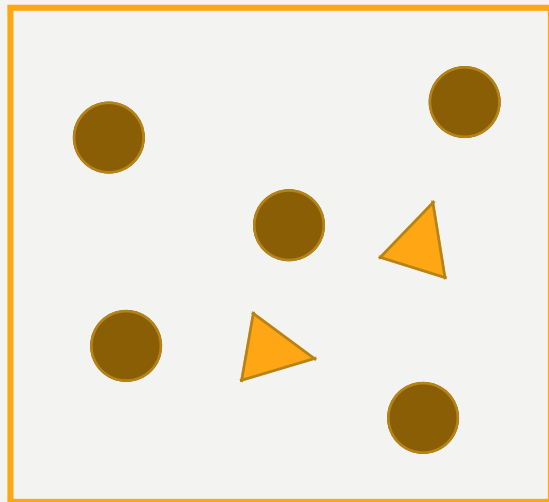
$$H = - \sum_{i=1}^K p_K \log_2 p_K$$

# ENTROPIA



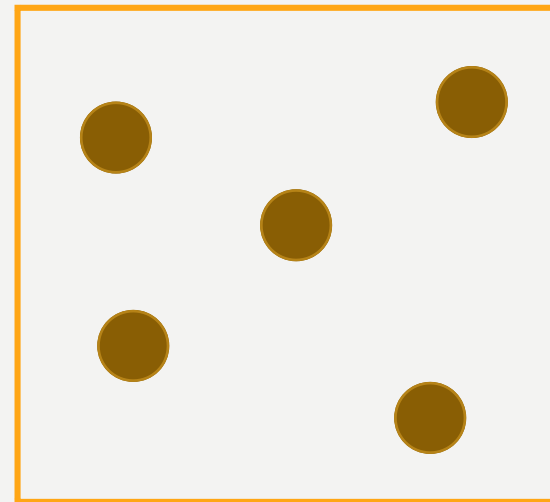
$$H = -\frac{5}{10}\log_2\frac{5}{10} - \frac{5}{10}\log_2\frac{5}{10}$$

$$H = 1$$



$$H = -\frac{5}{7}\log_2\frac{5}{7} - \frac{2}{7}\log_2\frac{2}{7}$$

$$H = 0,598$$



$$H = -\frac{5}{5}\log_2\frac{5}{5} - \frac{0}{5}\log_2\frac{0}{5}$$

$$H = 0$$

# GANHO DE INFORMAÇÃO

- Ganho de Informação mede a mudança de entropia que um atributo causa na classe, a variação de entropia.
- Essa medida nos mostra o quanto um atributo está correlacionado com a classe.

# GANHO DE INFORMAÇÃO

- A medida do Ganho de Informação é definida como:

$$\Delta H = H - \sum_{i=1}^K \frac{m_K}{m} \cdot H_K$$



# GANHO DE INFORMAÇÃO

## EXEMPLO

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	<i>No</i>
Sunny	Hot	High	True	<i>No</i>
Overcast	Hot	High	False	<i>Yes</i>
Rainy	Mild	High	False	<i>Yes</i>
Rainy	Cool	Normal	False	<i>Yes</i>
Rainy	Cool	Normal	True	<i>No</i>
Overcast	Cool	Normal	True	<i>Yes</i>
Sunny	Mild	High	False	<i>No</i>
Sunny	Cool	Normal	False	<i>Yes</i>
Rainy	Mild	Normal	False	<i>Yes</i>
Sunny	Mild	Normal	True	<i>Yes</i>
Overcast	Mild	High	True	<i>Yes</i>
Overcast	Hot	Normal	False	<i>Yes</i>
Rainy	Mild	High	True	<i>No</i>

Fonte: Information Gain\*

# GANHO DE INFORMAÇÃO

## EXEMPLO

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Fonte: Information Gain\*

Calculo da Entropia da Classe (H)

$$\begin{aligned} H(Y) &= - \sum_{i=1}^K p_k \log_2 p_k \\ &= - \frac{5}{14} \log_2 \frac{5}{14} - \frac{9}{14} \log_2 \frac{9}{14} \\ &= 0.94 \end{aligned}$$

# GANHO DE INFORMAÇÃO

## EXEMPLO

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Fonte: Information Gain\*

Cálculo do Ganho de Informação do atributo Humidade:

Cálculo da Probabilidade

$$\begin{aligned}
 \text{InfoGain}(\text{Humidity}) &= H(Y) - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R \\
 &= 0.94 - \frac{7}{14} H_L - \frac{7}{14} H_R
 \end{aligned}$$

# GANHO DE INFORMAÇÃO

## EXEMPLO

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Fonte: Information Gain\*

Cálculo do Ganho de Informação do atributo Humidade:

Cálculo das Entropias Condicionais para Valor de Atributo igual a **NORMAL**

$$H_L = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} \\ = 0.592$$

# GANHO DE INFORMAÇÃO

## EXEMPLO

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Fonte: Information Gain\*

Cálculo do Ganho de Informação do atributo Humidade:

Cálculo das Entropias Condicionais para Valor de Atributo igual a HIGH

$$H_R = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.985$$



# GANHO DE INFORMAÇÃO

## EXEMPLO

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Fonte: Information Gain\*

Ganho de Informação do Atributo Humidade:

$$\begin{aligned}
 \text{InfoGain}(\text{Humidity}) &= \\
 H(Y) - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R \\
 0.94 - \frac{7}{14} 0.592 - \frac{7}{14} 0.985 \\
 &= 0.94 - 0.296 - 0.4925 \\
 &= 0.1515
 \end{aligned}$$

# GANHO DE INFORMAÇÃO

## EXEMPLO

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Fonte: Information Gain\*

Ganho de Informação dos Atributos:

Outlook = 0.247

Temperature = 0.029

Humidity = 0.152

Windy = 0.048

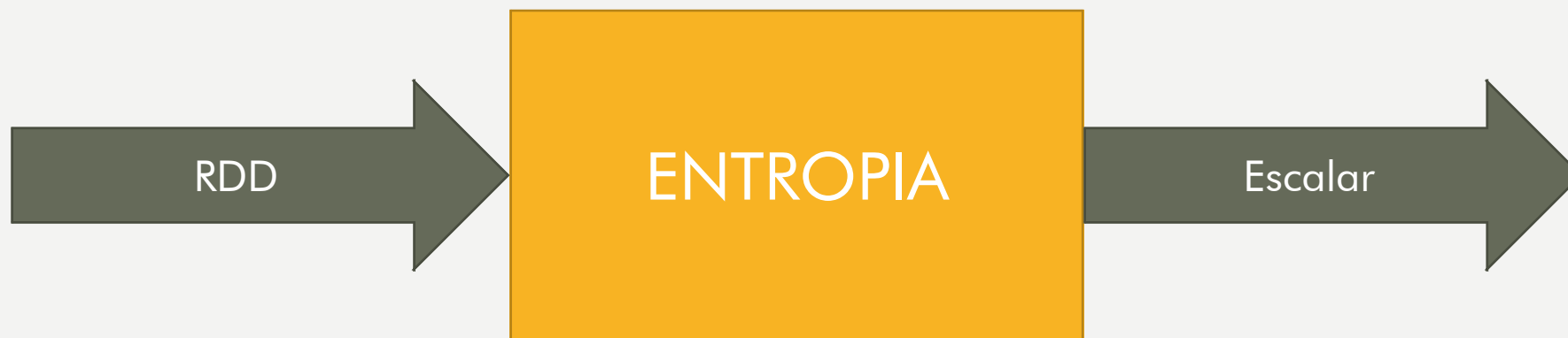


**CÓDIGO**



# CALCULO DE ENTROPIA

- Função que Calcula a Entropia de um conjunto de dados.
- Recebe um conjunto de dados (RDD) e retorna a entropia (escalar)



# CALCULO DE ENTROPIA

```
def Entropia (classe):  
    """Calcula a Entropia de Shannon de uma distribuição de dados.  
  
    Args:  
        classe (RDD): RDD contendo conjunto de dados a ser calculado a entropia.  
                     Valores devem ser categóricos.  
  
    Returns:  
        float: valor de Entropia de Shannon calculado para o RDD.  
    """  
    #counts calcula paralelamente o conteúdo do RDD  
    #como tuplas contendo (tipo, quantidade)  
    counts = (classe.map(lambda x: (x, 1)).reduceByKey(lambda a,b: a + b))  
    # n recebe o valor total de itens do RDD  
    n = classe.count()  
  
    # probs calcula a probabilidade de cada um dos estados do RDD  
    probs = counts.map(lambda x: x[1]/float(n))  
  
    # Entropia calcula a entropia do RDD  
    ## a função map faz o calculo da Entropia de cada um dos estados  
    ## a função reduce faz o somatório da entropia de Shannon  
    entropia = (probs.map(lambda p: -p*math.log(p,2)).reduce(lambda a,b: a + b))  
  
    # retorna valor escalar referente a entropia do RDD.  
    return entropia
```

# CALCULO DE ENTROPIA

- Conta valores do conjunto de dados

```
#counts calcula paralelamente o conteúdo do RDD  
#como tuplas contendo (tipo, quantidade)  
counts = (classe.map(lambda x: (x, 1))  
          .reduceByKey(lambda a,b: a + b))
```

# CALCULO DE ENTROPIA

- Conta número de elementos no conjunto de dados

```
# n recebe o valor total de itens do RDD  
n = classe.count()
```

# CALCULO DE ENTROPIA

- Calcula vetor de probabilidades para cada tipo de dado do conjunto de dados

```
# probs calcula a probabilidade de cada um dos estados do RDD  
probs = counts.map(lambda x: x[1]/float(n))
```

# CALCULO DE ENTROPIA

- Faz o cálculo da Entropia de Shannon

```
# Entropia calcula a entropia do RDD  
## a função map faz o calculo da Entropia de cada um dos estados  
## a função reduce faz o somatório da entropia de Shannon  
entropia = (probs.map(lambda p: -p*math.log(p,2))  
            .reduce(lambda a,b: a + b))
```

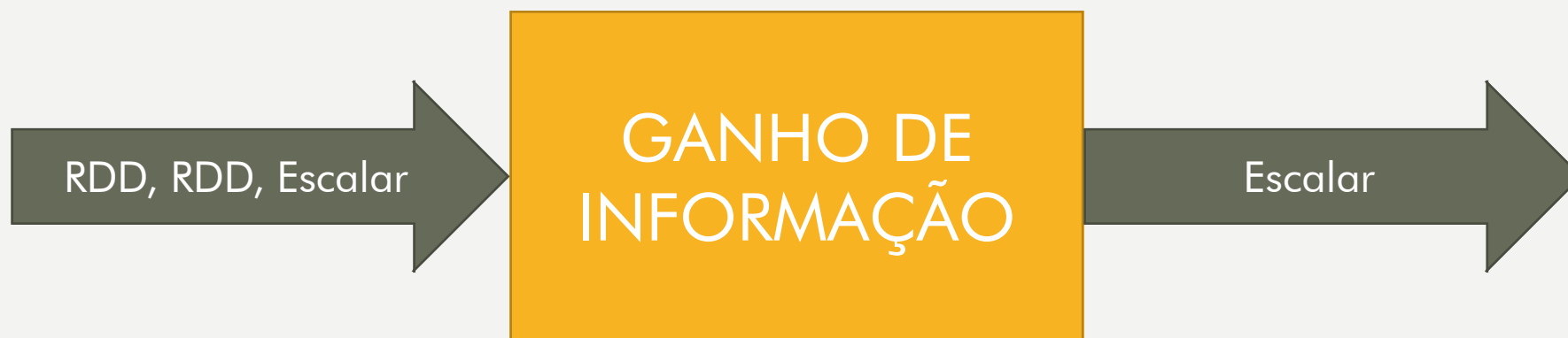
# CALCULO DE ENTROPIA

- Retorna escalar da Entropia

```
# retorna valor escalar referente a entropia do RDD.  
return entropia
```

# CALCULO DE GANHO DE INFORMAÇÃO

- Função que Calcula o Ganho de Informação de um atributo para uma classe.
- Recebe dois conjuntos de dados (RDD) para classe e atributo, e um Escalar referente ao valor de Entropia para a Classe; retorna o Ganho de Informação (Escalar)





# CALCULO DE GANHO DE INFORMAÇÃO

```
def infoGain (feature, classe, H):  
    """Calcula o ganho de informação de um atributo em relação a uma classe.  
  
    Args:  
        feature (RDD): RDD contendo os conjuntos de dados do atributo a ser  
            calculado o Ganho de Informação  
  
        classe (RDD): RDD contendo conjunto de dados da classe  
  
        H (float): Entropia da Classe, previamente calculada.  
  
    Returns:  
        float: valor de ganho de informação (redução da Entropia) que o atributo fornece sobre a classe  
    """  
    # calcula paralelamente o conteúdo do RDD  
    # como tuplas contendo (tipo, quantidade)  
    feat_count = feature.map(lambda x: (x, 1))\  
        .reduceByKey(lambda a,b: a + b)\  
        .collect()  
  
    # calcula as Entropias de um conjunto da classe dado cada um dos estados do atributo  
    entropiasN = [Entropia(classe.zip(feature)  
        .filter(lambda x: x[1]==v)  
        .map(lambda x: x[0])) for v,_ in feat_count]  
  
    # calcula a quantidade de itens no atributo  
    n = classe.count()  
  
    # calcula o ganho de informação do atributo.  
    ig = H - sum([(f[1]/float(n))*p for f,p in zip(feat_count, entropiasN)])  
  
    return ig
```

# CALCULO DE GANHO DE INFORMAÇÃO

- Conta valores do conjunto de dados

```
# calcula paralelamente o conteúdo do RDD  
# como tuplas contendo (tipo, quantidade)  
feat_count = feature.map(lambda x: (x, 1))\  
                    .reduceByKey(lambda a,b: a + b)\  
                    .collect()
```

- Dados cabem na memória, logo pode aplicar .collect()

# CALCULO DE GANHO DE INFORMAÇÃO

- Calcula cada uma das  $n$  Entropias da classe, filtrada pelo tipo de atributo, para cada um dos  $n$  valores que o atributo assume.

```
# calcula as Entropias de um conjunto da classe dado cada um dos estados do atributo  
entropiasN = [Entropia(classe.zip(feature)  
                      .filter(lambda x: x[1]==v)  
                      .map(lambda x: x[0])) for v,_ in feat_count]
```

# CALCULO DE GANHO DE INFORMAÇÃO

- Calcula quantidade de itens na classe.

```
# calcula a quantidade  
n = classe.count()
```

# CALCULO DE GANHO DE INFORMAÇÃO

- Calcula o Ganho de Informação

```
# calcula o ganho de informação do atributo.  
ig = H - sum([(f[1]/float(n))*p for f,p in zip(feat_count, entropiasN)])
```

- Nesse caso não é utilizado processamento paralelizado pois todas as variáveis já foram reduzidas.

# CALCULO DE GANHO DE INFORMAÇÃO


- Retorna Ganho de Informação

```
return ig
```

A decorative wavy line in yellow and white on the left side of the slide.

# BASE DE DADOS

2015 FLIGHT DELAYS AND  
CANCELLATIONS

 Reviewed Dataset

172

## 2015 Flight Delays and Cancellations

Which airline should you fly on to avoid significant delays?



Department of Transportation · last updated a year ago

[Overview](#)[Data](#)[Kernels](#)[Discussion](#)[Activity](#)

Download (192 MB)

[New Kernel](#)

Tags

aviation

medium

featured

Top Contributors



FabienDaniel

1st



DSEverything

2nd



user123

3rd

Kernels

[Predicting flight delays \[Tutorial\]](#)

175 votes · 7 months ago

[Airline Delays EDA, Deep Dive & Le...](#)

22 votes · 6 months ago

[Insights and ranking module of airli...](#)

6 votes · a year ago

Discussion

[Context of dataset?](#)

0 replies · 3 months ago

[Where is the data available throug...](#)

0 replies · 3 months ago

[NA Values for Delay Reasons](#)

0 replies · 4 months ago



# BANCO DE DADOS

- YEAR
- MONTH
- DAY
- DAY\_OF\_WEEK
- AIRLINE
- FLIGHT\_NUMBER
- TAIL\_NUMBER
- ORIGIN\_AIRPORT
- DESTINATION\_AIRPORT
- SCHEDULED\_DEPARTURE
- DEPARTURE\_TIME
- DEPARTURE\_DELAY
- TAXI\_OUT
- WHEELS\_OFF
- SCHEDULED\_TIME
- ELAPSED\_TIME
- AIR\_TIME
- DISTANCE
- WHEELS\_ON
- TAXI\_IN
- SCHEDULED\_ARRIVAL
- ARRIVAL\_TIME
- CANCELLED
- CANCELLATION\_REASON
- AIR\_SYSTEM\_DELAY
- SECURITY\_DELAY
- AIRLINE\_DELAY
- LATE\_AIRCRAFT\_DELAY
- WEATHER\_DELAY
- ARRIVAL\_DELAY
- DIVERTED

# ESCOLHIDOS

- YEAR
- MONTH
- DAY
- DAY\_OF\_WEEK
- AIRLINE
- FLIGHT\_NUMBER
- TAIL\_NUMBER
- ORIGIN\_AIRPORT
- DESTINATION\_AIRPORT
- SCHEDULED\_DEPARTURE
- DEPARTURE\_TIME
- DEPARTURE\_DELAY
- TAXI\_OUT
- WHEELS\_OFF
- SCHEDULED\_TIME
- ELAPSED\_TIME
- AIR\_TIME
- DISTANCE
- WHEELS\_ON
- TAXI\_IN
- SCHEDULED\_ARRIVAL
- ARRIVAL\_TIME
- CANCELLED
- CANCELLATION\_REASON
- AIR\_SYSTEM\_DELAY
- SECURITY\_DELAY
- AIRLINE\_DELAY
- LATE\_AIRCRAFT\_DELAY
- WEATHER\_DELAY
- ARRIVAL\_DELAY
- DIVERTED

# OBJETIVO

- Verificar qual dos atributos selecionados está mais fortemente correlacionado com o **Cancelamento de Voos Atrasados**

A decorative graphic on the left side of the slide consisting of two parallel, wavy vertical lines. The inner line is yellow and the outer line is white, both set against a dark brown background.

# APLICAÇÃO

# VALIDAÇÃO

Outlook = 0.247

Temperature = 0.029

Humidity = 0.152

Windy = 0.048

```
In [18]: H = Entropia (Play)
         print H
```

0.940285958671

```
In [43]: igOutlook = infoGain(Outlook, Play, H)
         print igOutlook
```

0.246749819774

```
In [44]: igTemperature = infoGain(Temperature, Play, H)
         print igTemperature
```

0.029222565659

```
In [45]: igHumidity = infoGain(Humidity, Play, H)
         print igHumidity
```

0.151835501362

```
In [46]: igWindy = infoGain(Windy, Play, H)
         print igWindy
```

0.0481270304083

# VERIFICAÇÃO DE MELHOR NÚMERO DE PARTIÇÕES

```
# carregar base de dados
numPartitions = 1
rawData = sc.textFile(fileName, numPartitions)
Cancelled = rawData.map(lambda x: x.split(",")[22])
featDayofWeek = rawData.map(lambda x: x.split(",")[4])

start_time = time.time()
H = Entropia(Cancelled)
print H
print("--- %s seconds ---" % (time.time() - start_time))

start_time = time.time()
igDayofWeek = infoGain(featDayofWeek, Cancelled, H)
print igDayofWeek
print("--- %s seconds ---" % (time.time() - start_time))

print "number of partition:", numPartitions
```

```
0.00425591582633
--- 31.870000124 seconds
1.24766472434e-05
--- 490.332000017 seconds
number of partition: 1
```

```
0.00425591582633
--- 32.4440000057 seconds
1.24766472434e-05
--- 489.858999968 seconds
number of partition: 8
```

```
0.00425591582633
--- 32.8069999218 seconds
1.24766472434e-05
--- 495.953000069 seconds
number of partition: 2
```

```
0.00425591582633
--- 37.8569998741 seconds
1.24766472434e-05
--- 594.200999975 seconds
number of partition: 16
```

```
0.00425591582633
--- 32.2950000763 seconds
1.24766472434e-05
--- 492.785000086 seconds
number of partition: 4
```

```
0.00425591582633
--- 68.2899999619 seconds
1.24766472434e-05
--- 815.203999996 seconds
number of partition: 32
```

```
0.00425591582633
--- 32.1840000153 seconds
1.24766472434e-05
--- 489.861999989 seconds
number of partition: 6
```

# VERIFICAÇÃO DE MELHOR NÚMERO DE PARTIÇÕES

```
# carregar base de dados
numPartitions = 1
rawData = sc.textFile(fileName, numPartitions)
Cancelled = rawData.map(lambda x: x.split(",")[22])
featDayofWeek = rawData.map(lambda x: x.split(",")[4])

start_time = time.time()
H = Entropia(Cancelled)
print H
print("--- %s seconds ---" % (time.time() - start_time))

start_time = time.time()
igDayofWeek = infoGain(featDayofWeek, Cancelled, H)
print igDayofWeek
print("--- %s seconds ---" % (time.time() - start_time))

print "number of partition:", numPartitions
```

```
0.00425591582633
--- 31.870000124 seconds
1.24766472434e-05
--- 490.332000017 seconds
number of partition: 1
```

```
0.00425591582633
--- 32.4440000057 seconds
1.24766472434e-05
--- 489.858999968 seconds
number of partition: 8
```

```
0.00425591582633
--- 32.8069999218 seconds
1.24766472434e-05
--- 495.953000069 seconds
number of partition: 2
```

```
0.00425591582633
--- 37.8569998741 seconds
1.24766472434e-05
--- 594.200999975 seconds
number of partition: 16
```

```
0.00425591582633
--- 32.2950000763 seconds
1.24766472434e-05
--- 492.785000086 seconds
number of partition: 4
```

```
0.00425591582633
--- 68.2899999619 seconds
1.24766472434e-05
--- 815.203999996 seconds
number of partition: 32
```

```
0.00425591582633
--- 32.1840000153 seconds
1.24766472434e-05
--- 489.861999989 seconds
number of partition: 6
```

# RESULTADOS

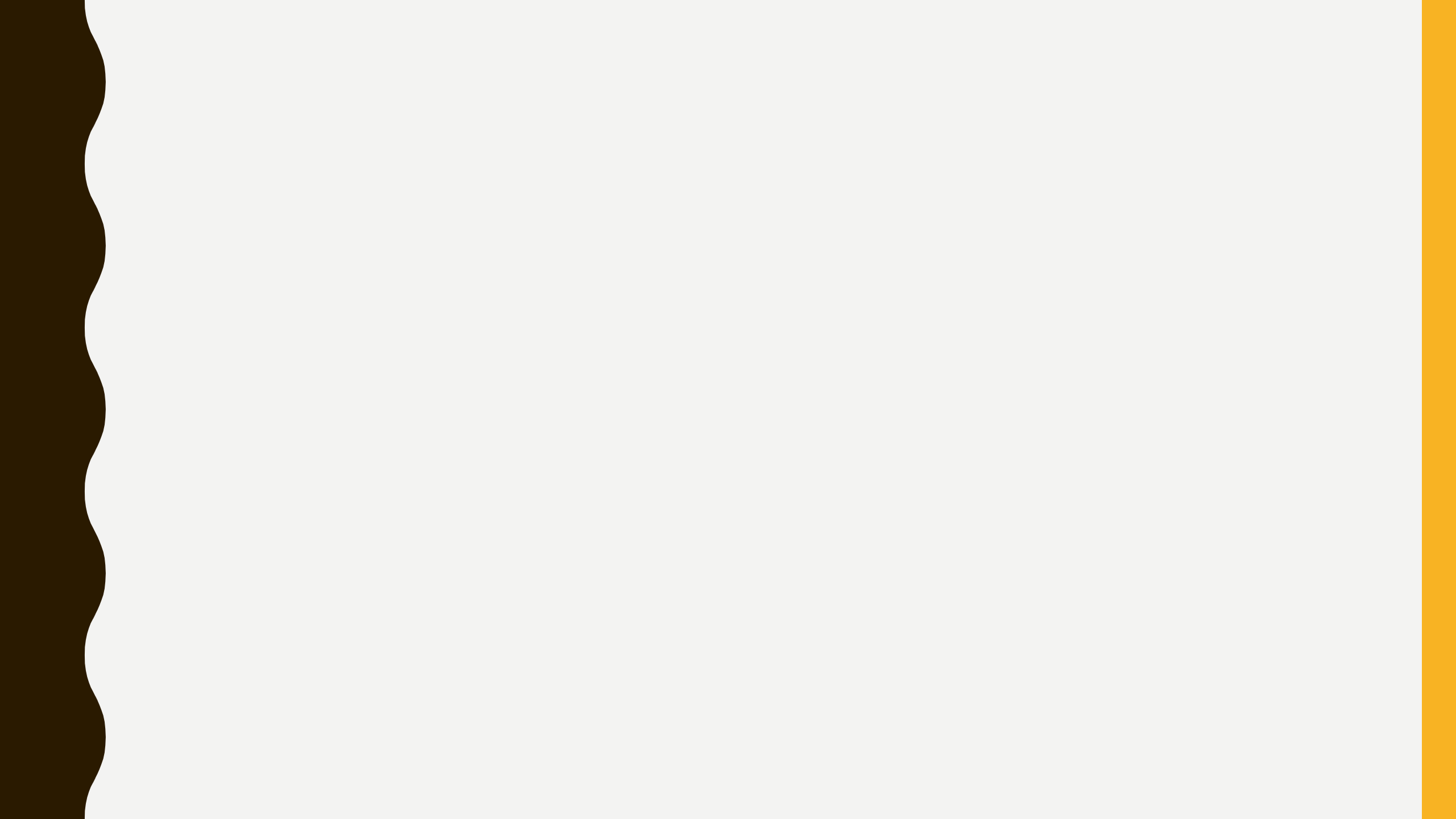
Tabela 1: Resultados

-	Mês	Dia do Mês	Dia da Semana	cia. Aérea
Ganho de Informação	0.0048013	0.0019261	0.0008652	0.0043092
Tempo com 8 Partições (min)	37.64	94.51	21.79	43.14
Tempo com 1 Partição (min)	36.46	90.98	21.26	41.94



# CONCLUSÃO

- O número de partições não interferiu no tempo de processamento do grupo de dados.
- Os atributos **Mês** em que o voo foi realizado e **Companhia Aérea** estão mais fortemente correlacionados com o **Cancelamento** dos voos atrasados.



# AGRADECIMENTOS

- Agradeço ao Prof. Dr. Fabricio Olivetti por ministrar o curso e me encorajar a continuá-lo até o fim.
- Agradeço ao Prof. Dr. Ronaldo Prati, meu coorientador no mestrado, pela ajuda na realização deste projeto.



**OBRIGADO**