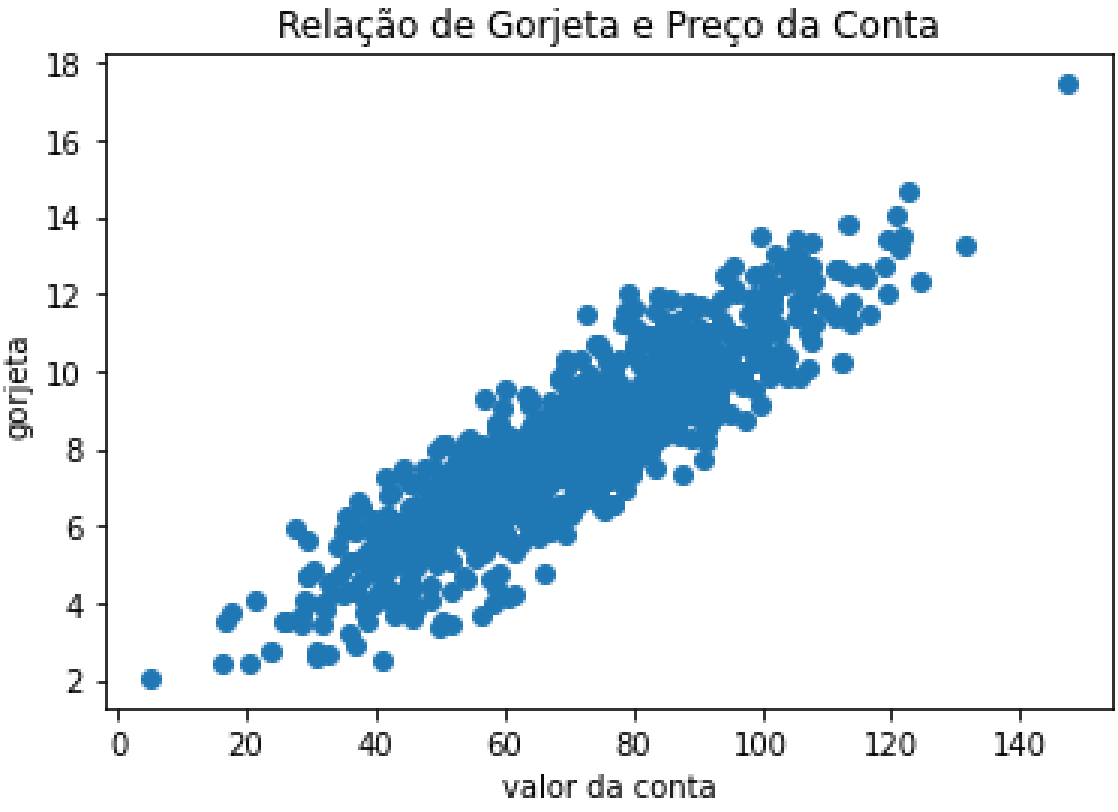
ÁRVORES DE DECISÃO

PROBLEMAS LINEARES E NÃO-LINEARS

Quando aprendemos, no começo do curso, sobre regressão linear vimos que esse modelo funciona muito bem quando temos uma certa linearidade dos nossos dados.

Por exemplo, suponha que você trabalha em um restaurante e quer estimar quanto de *gorjeta* os seus garçons vão receber com base na conta final do restaurante. **Logo, quanto maior a conta do restaurante, maior a gorjeta.**



É claro que tem uma relação linear entre as variáveis. Podemos confirmar isso ao computar a correlação linear entre as variáveis :)

```
from scipy.stats import pearsonrcorr, _ = pearsonr(data1,
data2)
print("Correlação de Pearson: %.3f" % corr)Correlação de
Pearson: 0.887
```

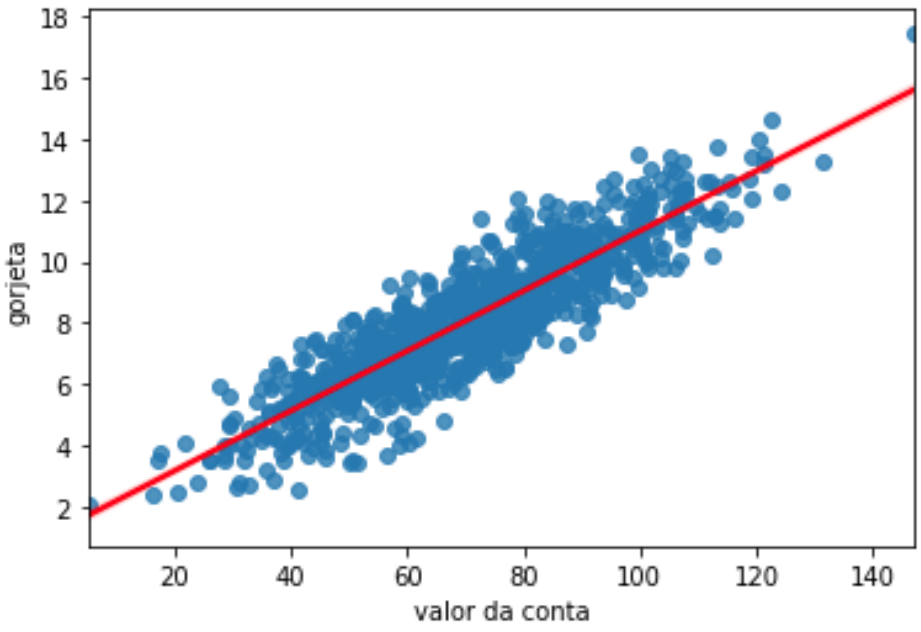
que 1 é uma correlação positiva perfeita e -1 é uma correlação negativa perfeita. Geralmente, um valor acima de 0.50 está associado com uma alta correlação positiva.

Vamos treinar um modelo de regressão para esse caso :)

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_scoreX_train, X_test, y_train,
y_test = train_test_split(
    data1.reshape(-1, 1), data2.reshape(-1, 1), test_size=0.1
)model = LinearRegression()
model.fit(X_train, y_train)
r2_score(y_test, model.predict(X_test))0.7238562525812482
```

Conseguimos um modelo de regressão com o r^2 muito bom :)

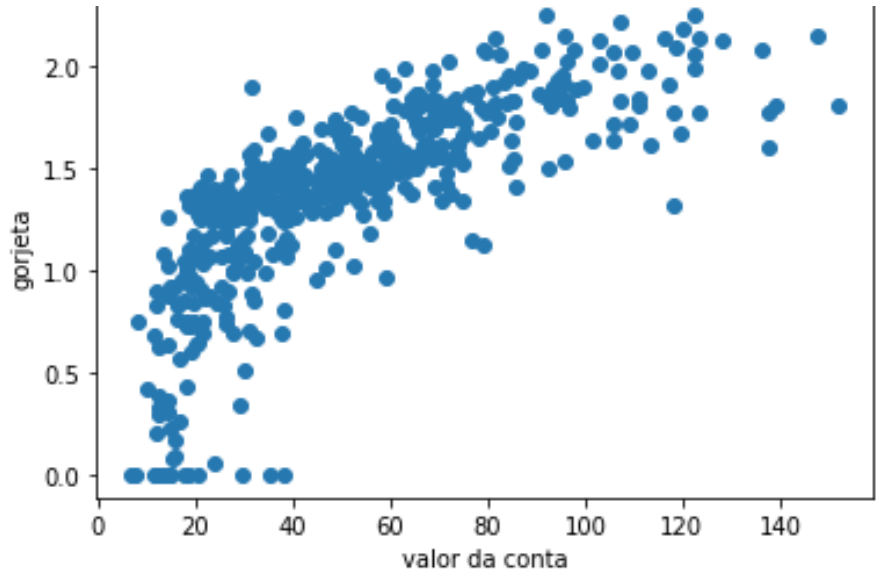
Podemos até fazer um plot da reta em relação aos dados:



Plot da reta aprendida pelo modelo

Contudo, vamos supor que **aconteceu uma doideira no restaurante por um tempo** e a relação das variáveis mudaram um pouco, seguindo o gráfico abaixo:

ÁRVORES DE DECISÃO



Uma **nova** relação entre o preço da conta e a gorjeta

Ou seja, a relação claramente deixou de ser claramente linear. Vamos computar a correlação de Pearson.

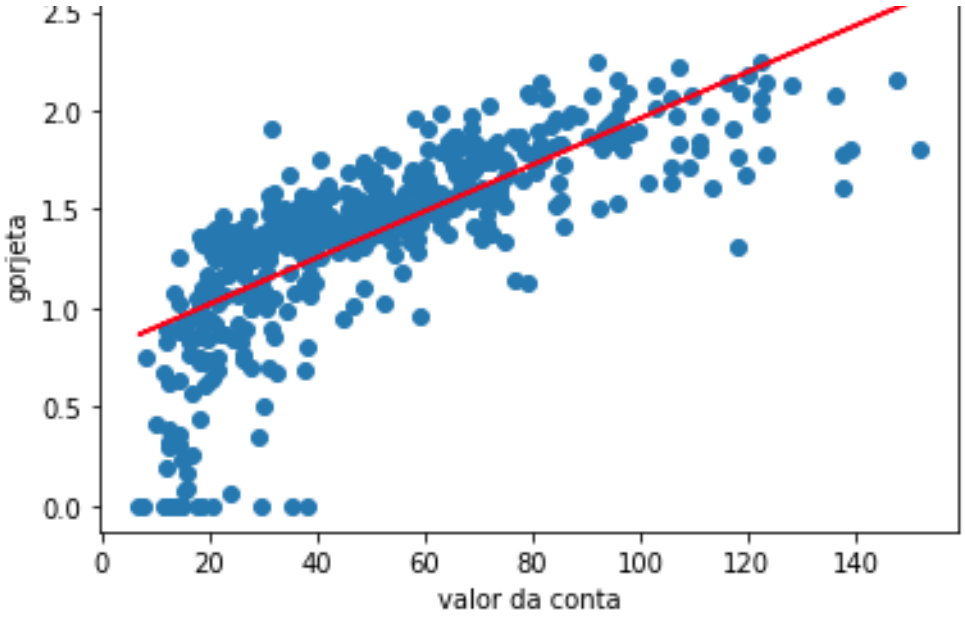
```
corr, _ = pearsonr(data1, data2)
print("Correlação de Pearson: %.3f" % corr)Correlação de
Pearson: 0.738
```

A correlação ainda é forte, mas é claramente menor do que a que calculamos inicialmente. Vamos ver o comportamento da regressão nesse caso?

```
X_train, X_test, y_train, y_test = train_test_split(
    data1.reshape(-1, 1), data2.reshape(-1, 1), test_size=0.1
)model = LinearRegression()
model.fit(X_train, y_train)
r2_score(y_test, model.predict(X_test))0.44411006406707687
```

Ou seja, claramente a regressão não foi muito boa. Vamos plotar a reta para esse caso:

ÁRVORES DE DECISÃO



Regressão linear aplicada para o novo conjunto de dados

Como a **regressão linear** resulta em uma **reta**, será que conseguimos fazer melhor?

AVANÇAR

ÁRVORES DE DECISÃO

INTRODUZINDO ÁRVORES DE DECISÃO

Árvores de Decisão são modelos não lineares e que, portanto, conseguem encontrar padrões que possuem esse tipo de relação entre as variáveis. Mas como ela funciona?

A ideia é relativamente simples. Uma árvore de decisão é construída ao **repartir** os dados seguindo regras de decisão. Cada dado, uma vez repartido, resulta em um subgrupo, definido como nó. Por exemplo:

Se o valor da conta for maior que 20 reais e menor do que 60, então o valor a gorjeta deve ser 1.

O ALGORITMO DAS ÁRVORES DE DECISÃO

Podemos definir um algoritmo de árvore de decisão seguindo o seguinte o código, inspirado no **link**:

```
def monta_arvore(L):
 cria um nó t
 se o critério de parada é True:
 defina um modelo preditivo para t
 else:
 encontre o melhor split binário L = L_left + L_right
 t.left = monta_arvore(L_left)
 t.right = monta_arvore(L_right)
 return t
```

Mas como que essas regras são definidas?

REGRESSION TREES

No caso de árvores de decisão aplicadas a problemas de regressão, o modelo busca **partir os dados** seguindo um critério de quebra. No caso, queremos agrupar os nossos dados de forma que esses dados resultem em um **grupo que minimizem o erro quadrático médio**. Relembrando a métrica, MSE, é definida por:

ÁRVORES DE DECISÃO

$$\frac{1}{N}\sum(y_{true} - y_{pred})^2$$

Métrica MSE

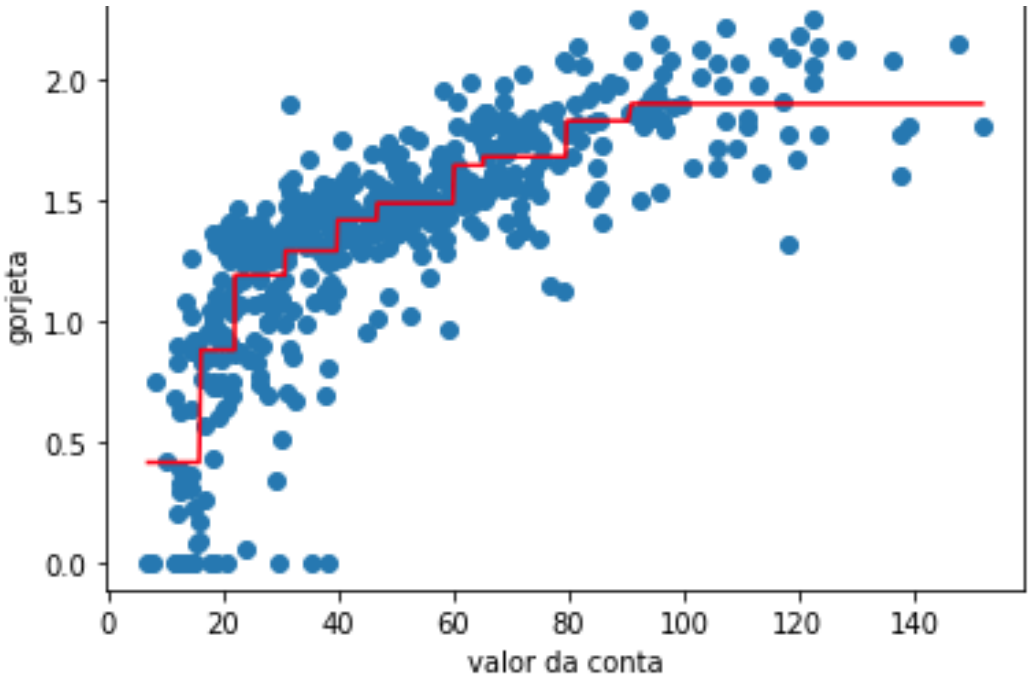
Ou seja, queremos minimizar a média dos erros elevado ao quadrado!

Vamos ao código para ilustrar isso:

```
from sklearn.tree import DecisionTreeRegressormodel =
DecisionTreeRegressor(criterion="mse", max_depth=4,
min_samples_leaf = 20)
model.fit(X_train, y_train)
r2_score(y_test, model.predict(X_test))0.5566920750334438
```

Fazendo o plot da “reta”:

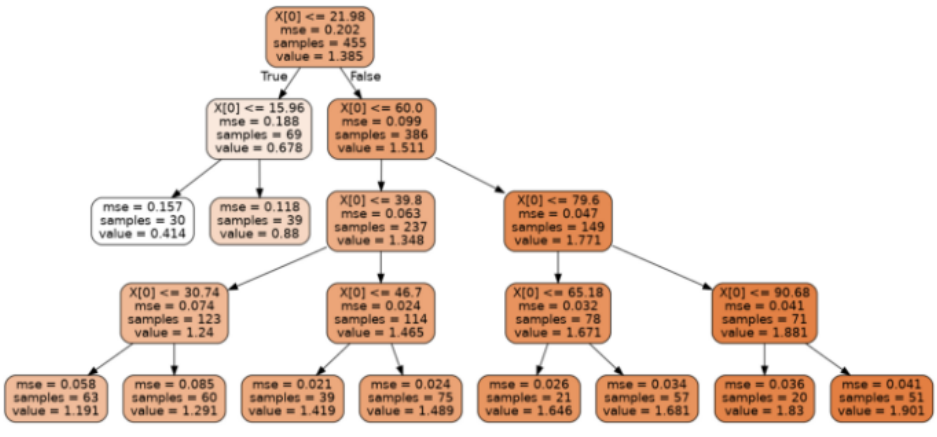
ÁRVORES DE DECISÃO



Plot da curva pela árvore de decisão

Ou seja, **claramente** o valor aprendido **não segue uma reta**, mas sim regras mais “secas” de corte dos nossos dados.

mport cv2 from pydotplus.graphviz import graph_from_dot_data from sklearn.tree import export_graphvizdot_data = export_graphviz(model, filled=True, rounded=True, out_file=None) graph = graph_from_dot_data(dot_data) graph.write_png('tree.png')img = cv2.imread('tree.png') plt.figure(figsize = (20, 20)) _ = plt.imshow(img)



Plot da árvore no caso de regressão

Nós podemos interpretar o valor acima da seguinte forma:

- Se o valor do preço da conta for **maior** que 21.98, **maior** que 60.0 e **menor que** 79.6, **menor do que** 65.18, **então** o preço da gorjeta, baseado na média de 21 amostras, deve ser 1.646

DESAFIOS DE ÁRVORE DE DECISÃO

Claramente, poderíamos construir uma árvore de decisão em que os **nós** possuíssem **apenas** uma **única amostra**. Contudo, isso **claramente** resultaria em árvore *overfitada* e *esse* é o maior problema das árvores de decisão: **elas overfittam** muito fácil e por conta disso eu coloquei alguns **hiperparâmetros** quando instanciei as classe *DecisionTreeRegressor*. Além do critério da árvore, os principais hiperparâmetros das árvores de decisão são:

- max_depth- a profundidade máxima da árvore (no nosso exemplo, usamos 3)
- max_features — o número **máximo** de features que o modelo olha para escolher a melhor regra de decisão
- min_samples_leaf — o número mínimo de amostras que queremos ter na árvore. No caso, usamos 20.

AVANÇAR

ÁRVORES DE DECISÃO

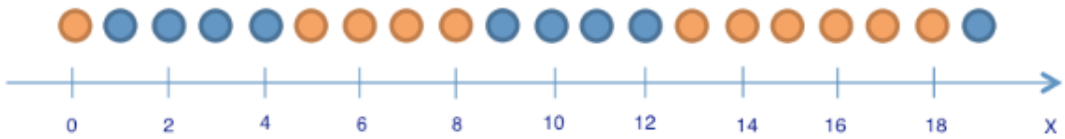
CLASSIFICATION TREES

*Inspirado fortemente no **link***

Como **Árvores de Decisão** funcionam no caso de problemas de classificação?

A primeira coisa a se pensar, contudo é que, como **estamos lidando com um problema de classificação**, **não** podemos ter *MSE* como critério de divisão. Para definirmos o critério ideal, vamos pegar o seguinte problema de classificação:

Suponha que você queira prever a cor de uma bola baseada na sua posição:



Nesse exemplo, temos 9 bolas azuis e 11 bolas amarelas. Se aleatoriamente pegarmos uma bola, ela será azul com a probabilidade 9/20 e amarela com probabilidade 11/20. Com isso, conseguimos ver que a chance de pegar uma bola de cada cor *é mais ou menos* a mesma. A partir dessa ideia, precisamos definir um conceito **novo**, chamado **entropia**.

ENTROPIA DE SHANNON

A entropia de Shannon é um conceito da **teoria da informação** que define o grau de **caos** de um sistema. Quanto maior a entropia, menos ordenado é um ambiente. Ela é uma métrica que é definida da seguinte forma:

ÁRVORES DE DECISÃO

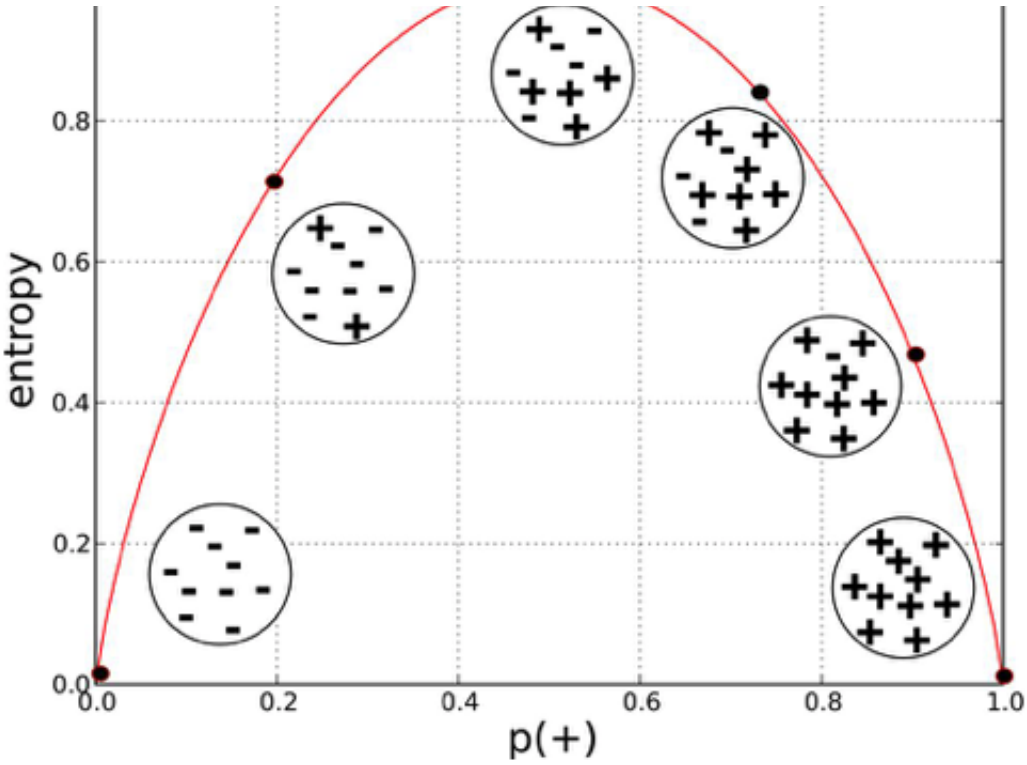
$$E = -\sum_{i}^{N} p_i log_2 p_i$$

Entropia de Shannon

O resultado da entropia é medido em *bits* e pode ser um valor entre 0 e 1, em que 0 seria um sistema bem ordenado, com *alto grau de certeza* e 1, um *alto grau de incerteza*. O **log** aqui é na base do **número de classes que estamos usando**. Como no nosso caso, estamos lidando com 2 classes, o log é na base 2.

Graficamente, ela pode ser ilustrada pelo gráfico abaixo:

ÁRVORES DE DECISÃO



Ou seja, segundo o exemplo, quando a probabilidade de pertencer a classe positva é 1 (apenas exemplos positivos), ou 0 (apenas exemplos negativos), a entropia resultante é 0, porquê **existe uma alta certeza da organização do ambiente**.

Disso, conseguimos derivar outra métrica, que é o ganho de informação:

$$IG(Y,X)=E(X)-E(Y|X)$$

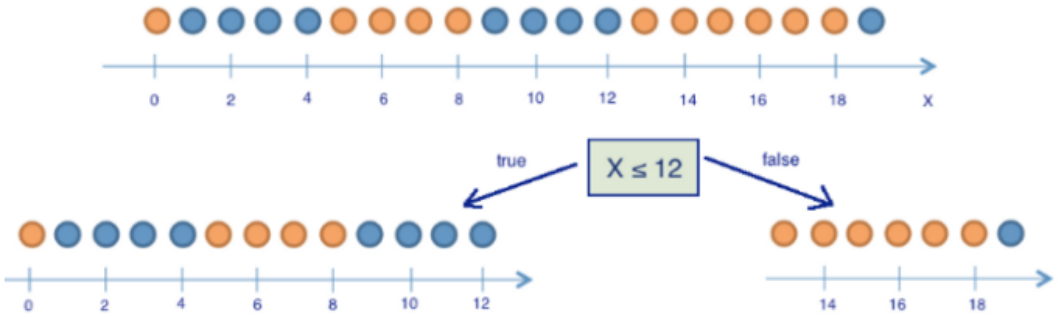
O ganho de informação de Y para X é dado pela Entropia de X menos a Entropia de Y dado X. **Na prática, quanto maior redução de incertezas, maior o ganho de informação possível**. Ou seja, existe uma relação **inversa** entre a entropia e o ganho de informação.

No caso, a entropia do sistema que estamos lidando é dado por:

ÁRVORES DE DECISÃO

$$H_0 = -(\frac{-}{20}log_2\frac{-}{20} + \frac{--}{20}log_2\frac{--}{20}) \approx 1$$

Entropia de Shannon para o exemplo



fonte

Logo, o nosso Y é basicamente dado pelo sistema pela regra de decisão Y ≤ 12. E X é o sistema todo, sem nenhuma regra de quebra até então (E(X)=H_0).

O grupo da esquerda tem 13 bolas, sendo 8 azuis e 5 amarelas, tendo como entropia H_1 e o grupo da direita tem 7 bolas, sendo 6 amarelas e 1 azul, tendo a entropia H_2.

$$H_1 = -(\frac{8}{13}log_2\frac{8}{13} + \frac{5}{13}log_2\frac{5}{13})$$

$$H_2 = -(\frac{6}{7}log_2\frac{6}{7} + \frac{1}{7}log_2\frac{1}{7})$$

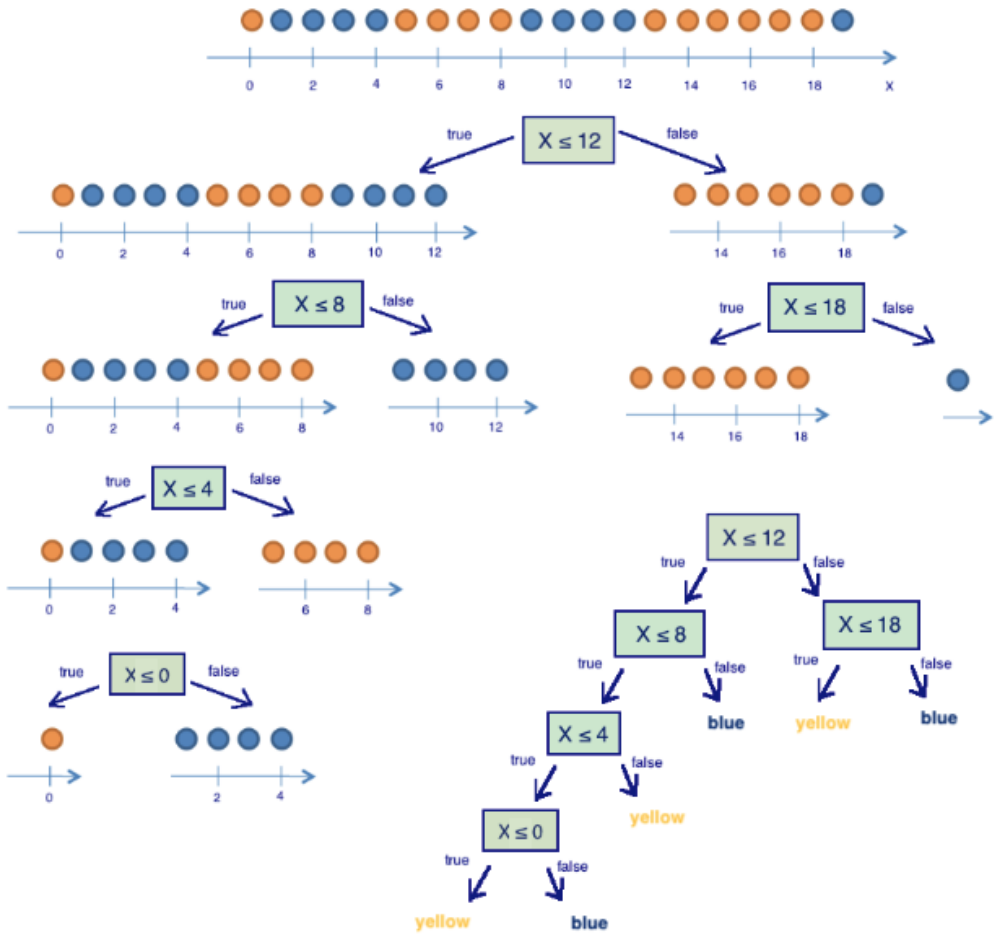
Então, podemos calcular o ganho de informação dessa parte do nosso dataset da seguinte forma:

ÁRVORES DE DECISÃO

$$IG(X \leq 12) = H_0 - \frac{20}{20} H_1 - \frac{20}{20} H_2 \approx 0.10$$

Ou seja, o uso da regra de decisão $X \leq 12$ resulta em um sistema mais ordenado que o anterior, uma vez que existem algum ganho de informação.

Repetimos, então, o processo:



Montagem inteira da árvore de decisão

Para o grupo da direita, a gente precisa apenas fazer uma partição a mais, uma vez que temos apenas uma bola na cor azul (basta perguntar se a posição da bola é menor ou igual que 18). Contudo, para o grupo da esquerda, precisamos repetir o processo três vezes. **Além disso, também é valido dizer que a entropia de um grupo onde todas as bolas são da mesma cor é 0**

ÁRVORES DE DECISÃO

Exemplo quando da entropia quando a probabilidade é 1

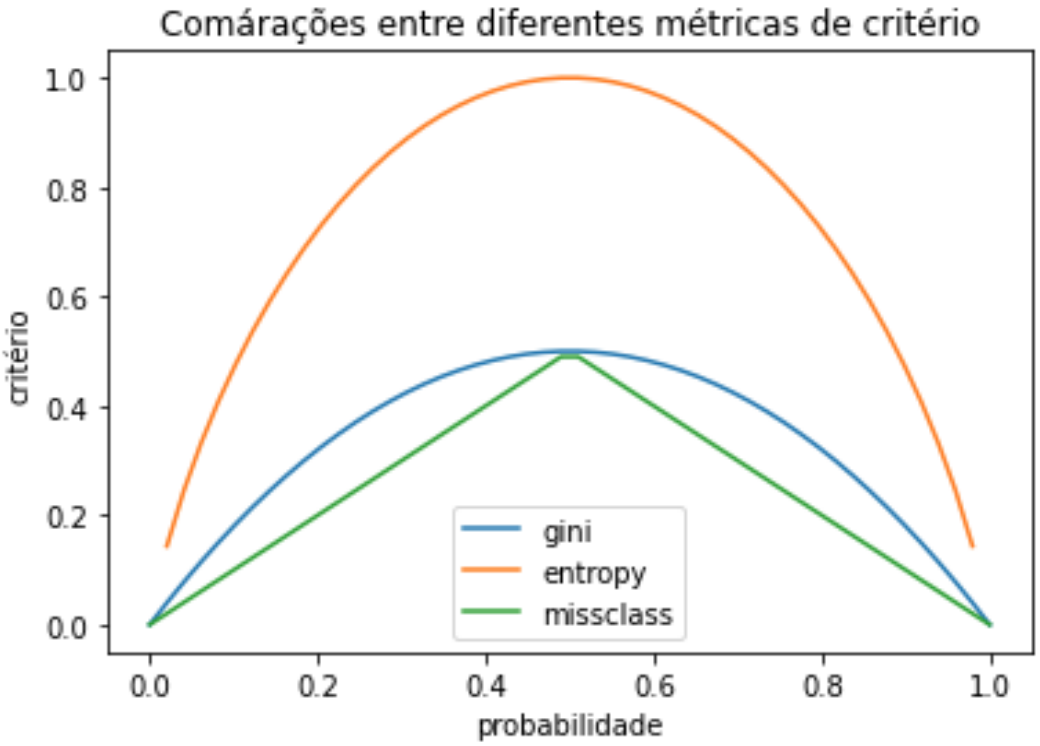
AVANÇAR

ÁRVORES DE DECISÃO

SÓ PODEMOS USAR A ENTROPIA?

Não!

Assim como na árvore para regressão, existem outros critérios de definição além da entropia, como o **Indice de Gini** . A idéia é que a relação entre as métricas seja parecida e isso pode ser visto com facilidade pelo gráfico abaixo:



Comparação entre métricas

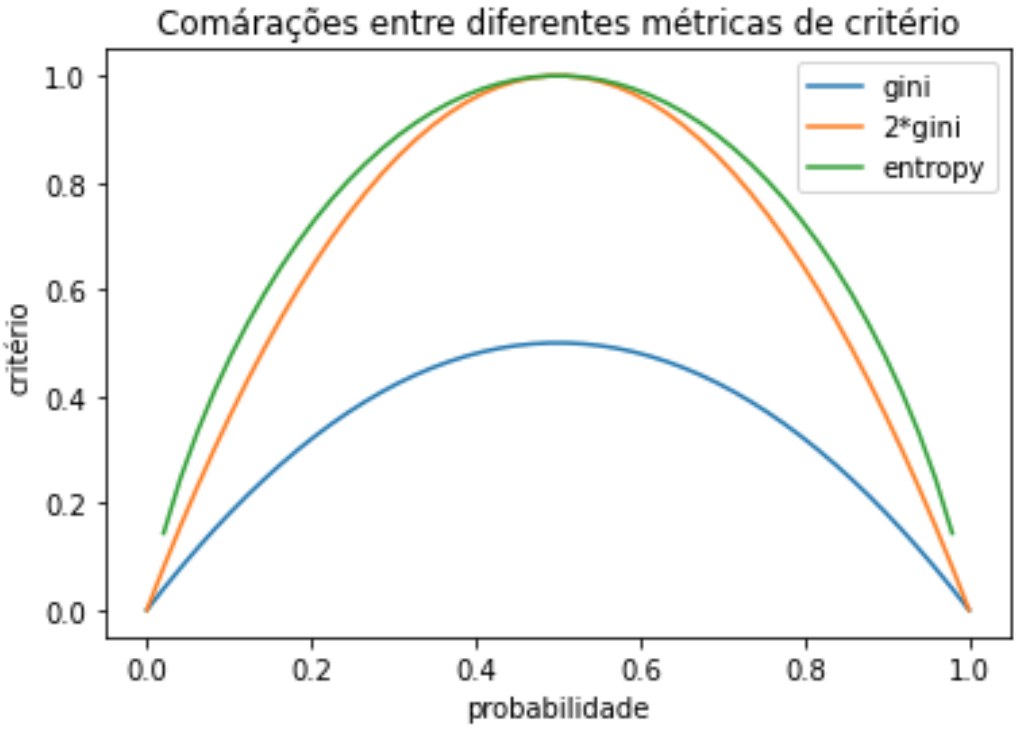
Em que o Gini e o missclassification são computadas, respectivamente por:

ÁRVORES DE DECISÃO

$$1 - \sum_i (p_i)^2$$
$$1 - \max_i p_i$$

Como computar, respectivamente o gini e a entropia

Na prática, as diferentes métricas (tendem a convertir para o mesmo resultado). A única diferença mais "prática" é que o índice de Gini é computacionalmente mais eficiente do que a entropia por não ter o cálculo do log. Além disso, se multiplicarmos o índice de Gini por 2, temos uma curva praticamente idêntica à entropia.

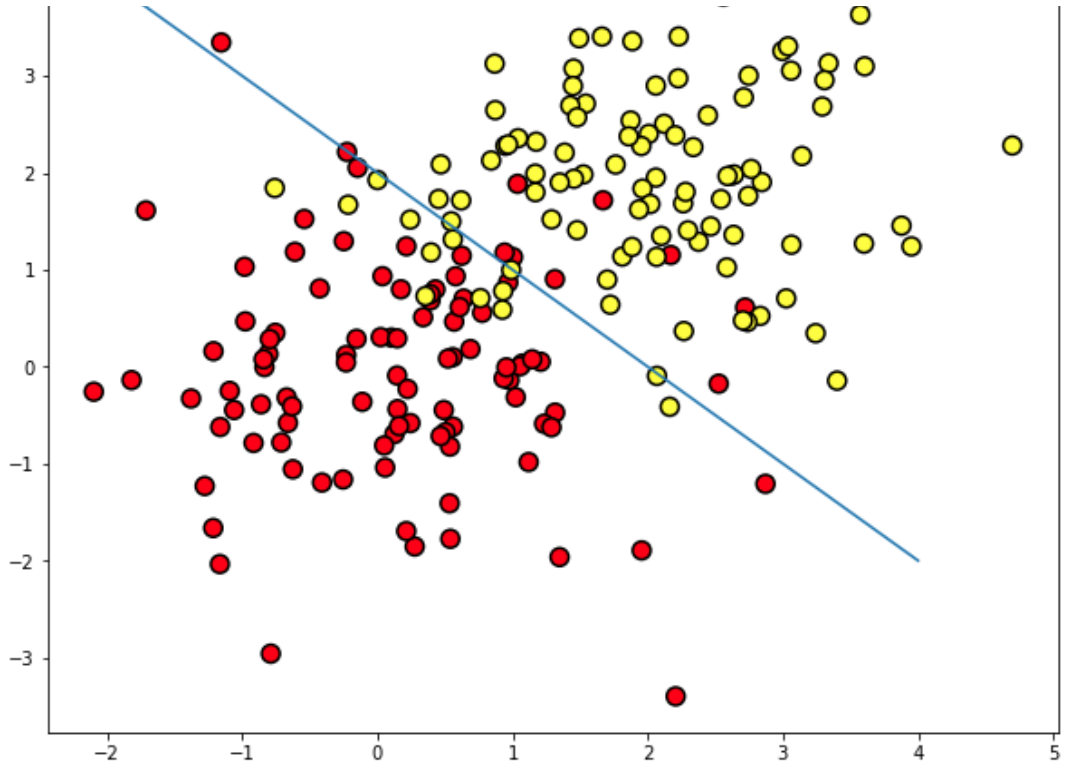


Comparação entre o Gini e o Entropy

APLICANDO UM EXEMPLO

Suponha que queremos criar um modelo que separe corretamente as classes abaixo:

ÁRVORES DE DECISÃO

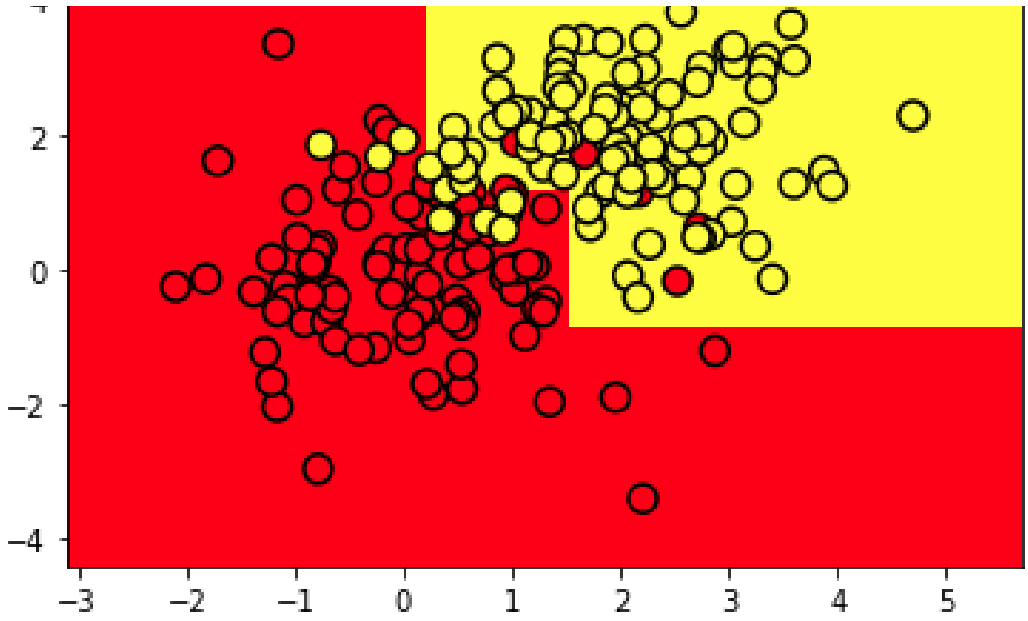


Exemplo de classificação simples

```
from sklearn.tree import DecisionTreeClassifierclf_tree =
DecisionTreeClassifier(criterion='entropy', max_depth=3)#
training the tree
clf_tree.fit(train_data, train_labels)
```

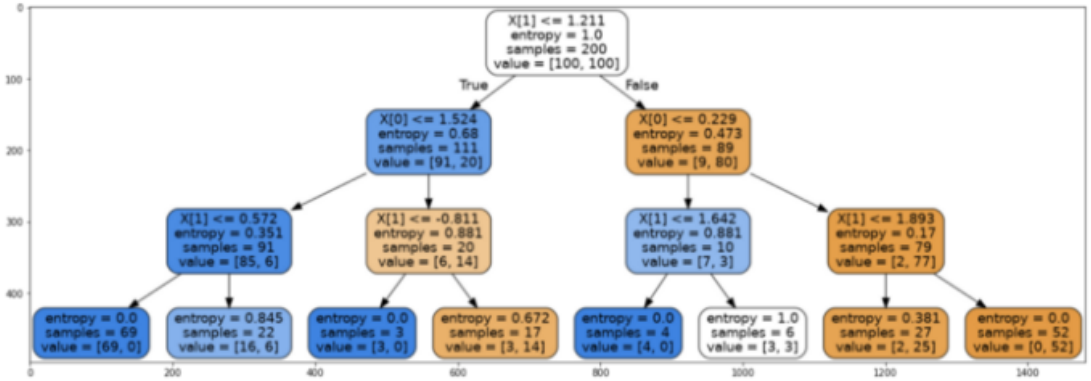
Podemos imprimir a **fronteira de decisão**.

ÁRVORES DE DECISÃO



Fronteira de decisão

```
dot_data = export_graphviz(
    clf_tree, filled=True, rounded=True,
    out_file=None
)
graph = graph_from_dot_data(dot_data)
graph.write_png('tree_classifier.png')img =
cv2.imread('tree_classifier.png')
plt.figure(figsize = (20, 20))
_ = plt.imshow(img)
```



Plot da árvore quando no caso de um problema de classificação

AVANÇAR

ÁRVORES DE DECISÃO

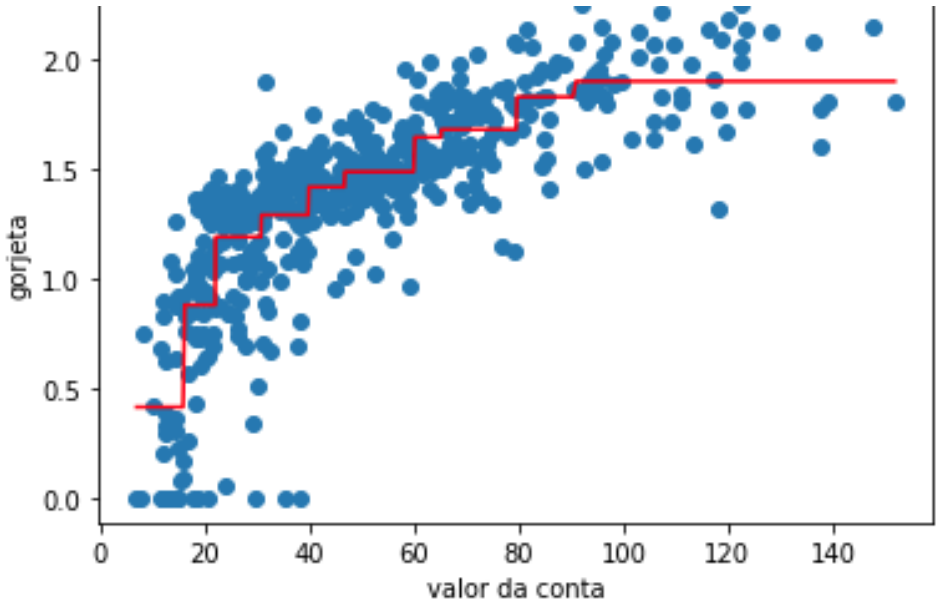
ÁRVORES DE DECISÃO

UM GOSTINHO DO QUE VEM POR AÍ

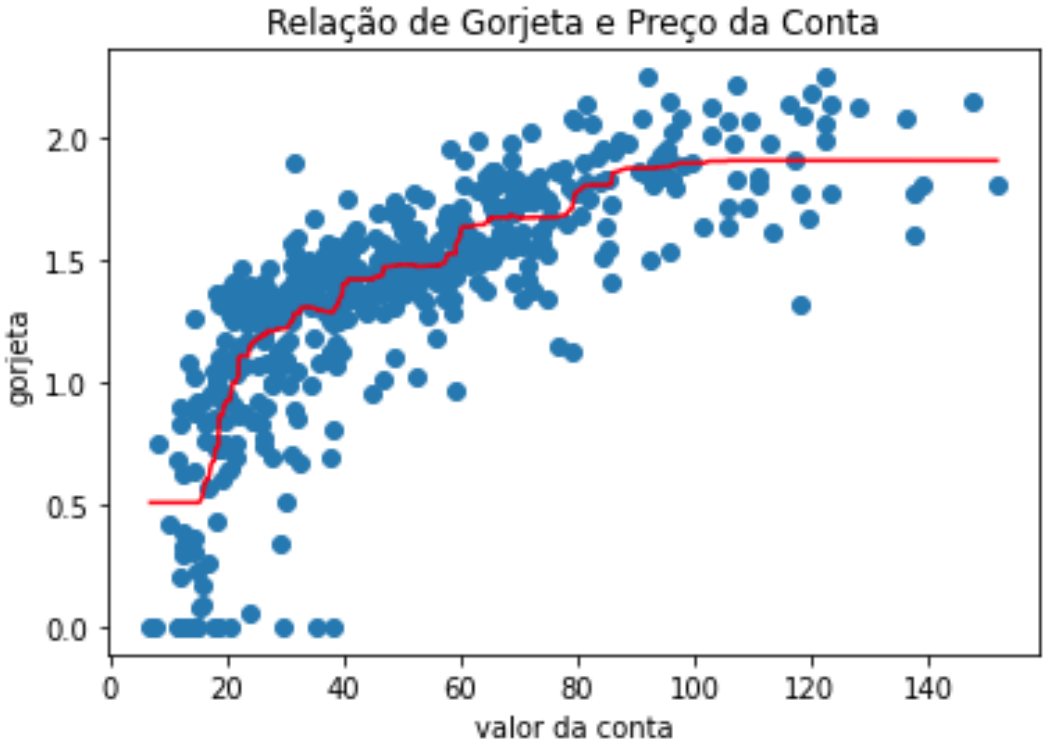
O interessante das Árvores de Decisão é que eles são a base para muitos algoritmos populares, como **Random Forests** e **Gradient Boosting**, que, por sua vez, são bastante utilizados na prática.

O interessante desses algoritmos é que eles são capazes de capturar não linearidades de uma maneira melhor. Pegando o exemplo da gorjeta, que tal compararmos uma árvore de decisão comum (o exemplo no começo do post) e uma random forest.

ÁRVORES DE DECISÃO



Plot da previsão com uma Árvore de Decisão



Plot da previsão com uma Random Forest

AVANÇAR

ÁRVORES DE DECISÃO

ÁRVORES DE DECISÃO

CONCLUSÃO

Parabéns! Vocês acabaram de aprender um pouco mais sobre um dos modelos mais clássicos de machine learning, as **Decision Trees** :)

REFERÊNCIAS

- https://towardsdatascience.com/https-medium-com-lorrli-classification-and-regression-analysis-with-decision-trees-c43cdbc58054

- https://mlcourse.ai/articles/topic3-dt-knn/#2.-Decision-Tree

- https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8

- https://scikit-learn.org/stable/modules/tree.html

O QUE ACHOU DESTA AULA?

Deixe seu feedback para continuarmos melhorando sua experiência.

1 MIN

AVALIAR

ÁRVORES DE DECISÃO