

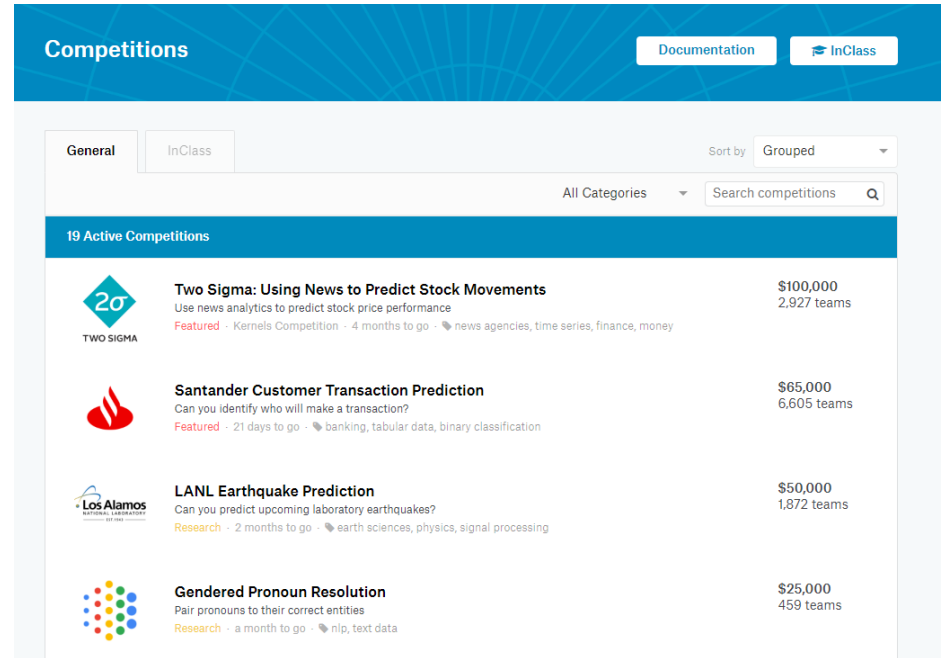
# MÉTODOS DE BOOSTING APLICADOS A DADOS TABULARES







Luiz Antônio Nonenmacher Júnior

# Motivação

# kaggle



The screenshot displays the Kaggle Competitions interface. At the top, there's a blue header with the word "Competitions" and buttons for "Documentation" and "InClass". Below the header, there are tabs for "General" and "InClass", and a "Sort by" dropdown set to "Grouped". A search bar labeled "Search competitions" is also present. The main section, titled "19 Active Competitions", lists four competitions:

Logo	Competition Name	Prize	Teams
	<b>Two Sigma: Using News to Predict Stock Movements</b> Use news analytics to predict stock price performance <i>Featured</i> · Kernels Competition · 4 months to go · news agencies, time series, finance, money	\$100,000	2,927 teams
	<b>Santander Customer Transaction Prediction</b> Can you identify who will make a transaction? <i>Featured</i> · 21 days to go · banking, tabular data, binary classification	\$65,000	6,605 teams
	<b>LANL Earthquake Prediction</b> Can you predict upcoming laboratory earthquakes? <i>Research</i> · 2 months to go · earth sciences, physics, signal processing	\$50,000	1,872 teams
	<b>Gendered Pronoun Resolution</b> Pair pronouns to their correct entities <i>Research</i> · a month to go · nlp, text data	\$25,000	459 teams

# Motivação



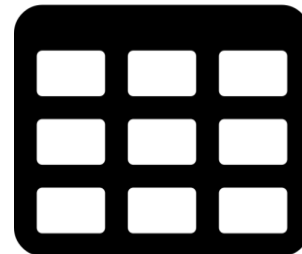
Textos



Imagens



Sons

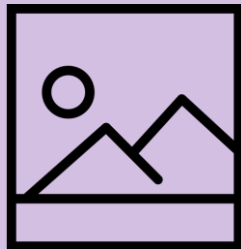


Tabelas

# Motivação



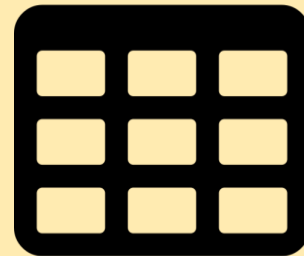
Textos



Imagens



Sons



Tabelas

Redes Neurais / Deep Learning

Gradient Boosted Trees  
*(XGBoost, LightGBM, CatBoost)*

# Contexto

Gradient Boosted Trees

# Contexto

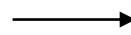
Gradient Boosted **Trees**



**Decision Trees**

# Contexto

Gradient Boosted Trees



Decision Trees



Gradient Boosting



Boosting



Ensembling

# Contexto

- **Decision Trees**
- **Ensembling**
  - Voting
  - Bagging
    - Random Forests
  - Boosting
    - Adaptative Boosting
    - Gradient Boosting
      - XGBoost, LightGBM, CatBoost
- **Implementação**

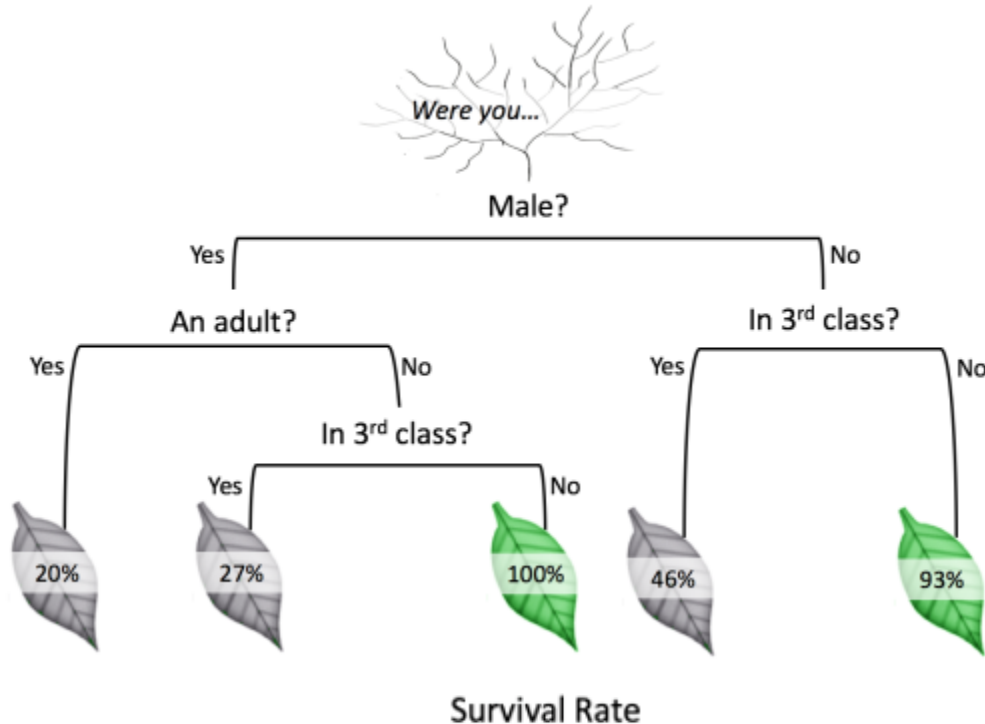


# Akinator

# Decision Trees



# Decision Trees

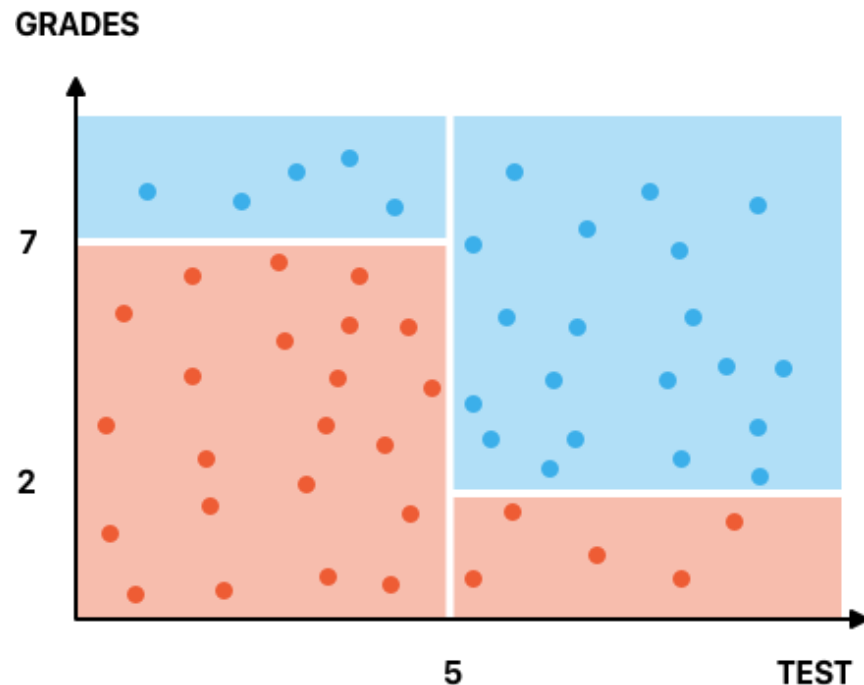
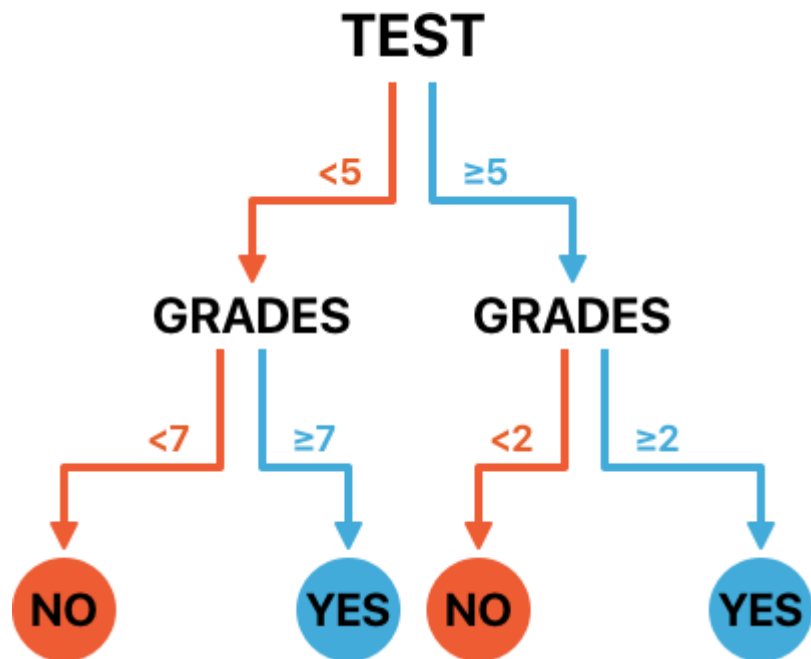


Uma árvore de decisão é montada através de uma sequencia de perguntas binárias que resultam em uma classe, uma probabilidade ou algum valor contínuo.

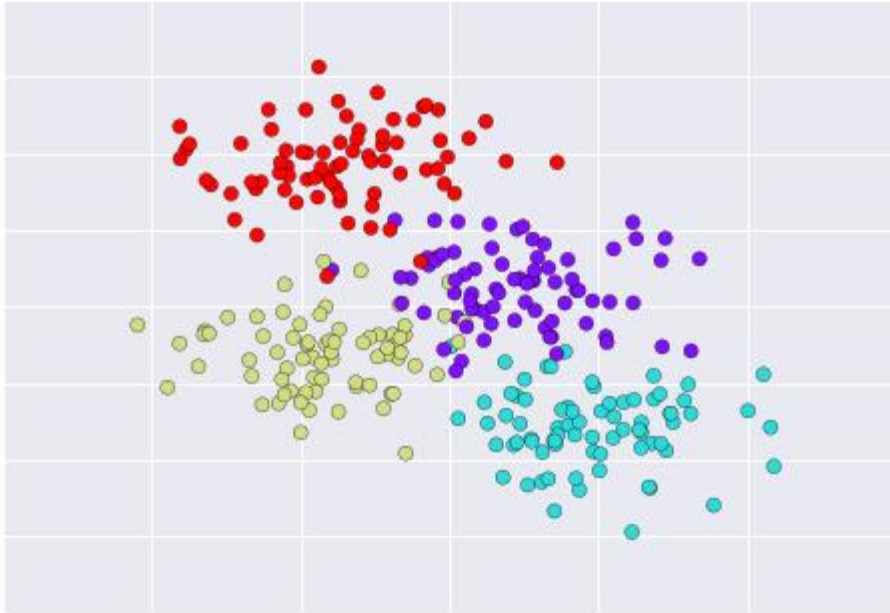
As perguntas são obtidas buscando maximizar o ganho de informação.

Uma forma de se visualizar árvores de decisão é que cada gera uma reta dividindo os pontos.

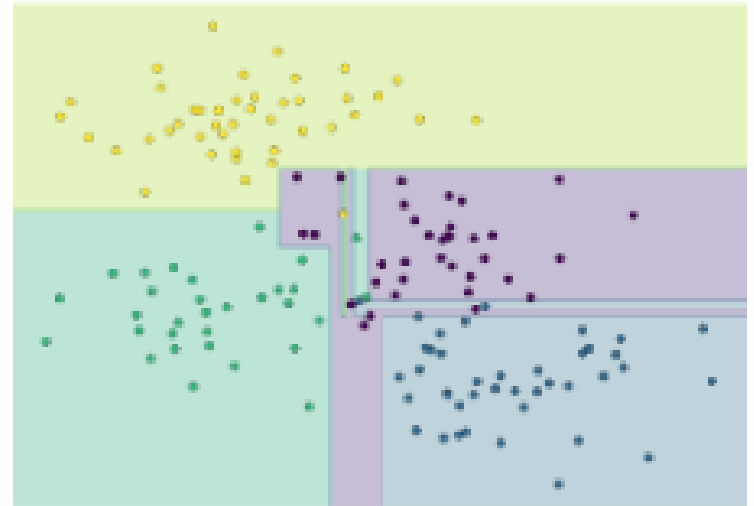
# Decision Trees



Dependendo do caso, árvores podem gerar regiões muito específicas e causas overfitting.



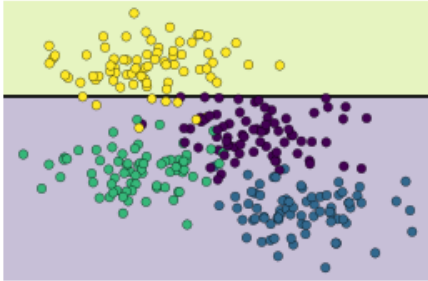
# Decision Trees



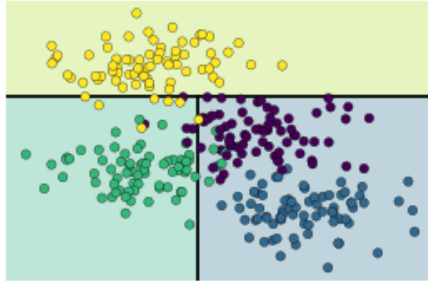
# Decision Trees

Uma das formas de contornar esse problema é através dos hiperparâmetros do modelo, como profundidade (depth) ou mínimo de elementos por nó.

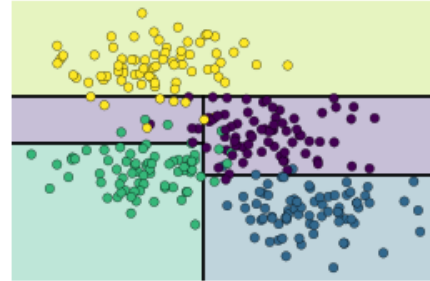
depth = 1



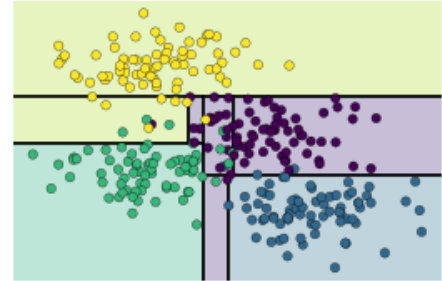
depth = 2



depth = 3



depth = 4



# Ensemble Learning

# Ensemble Learning

Combinação de diferentes modelos para obter um modelo final com desempenho melhor que os modelos individuais.

- Voting
- Bagging
- Boosting



# Voting



Imagine que treinamos cinco modelos para diferenciar cachorros de lobos.

# Voting



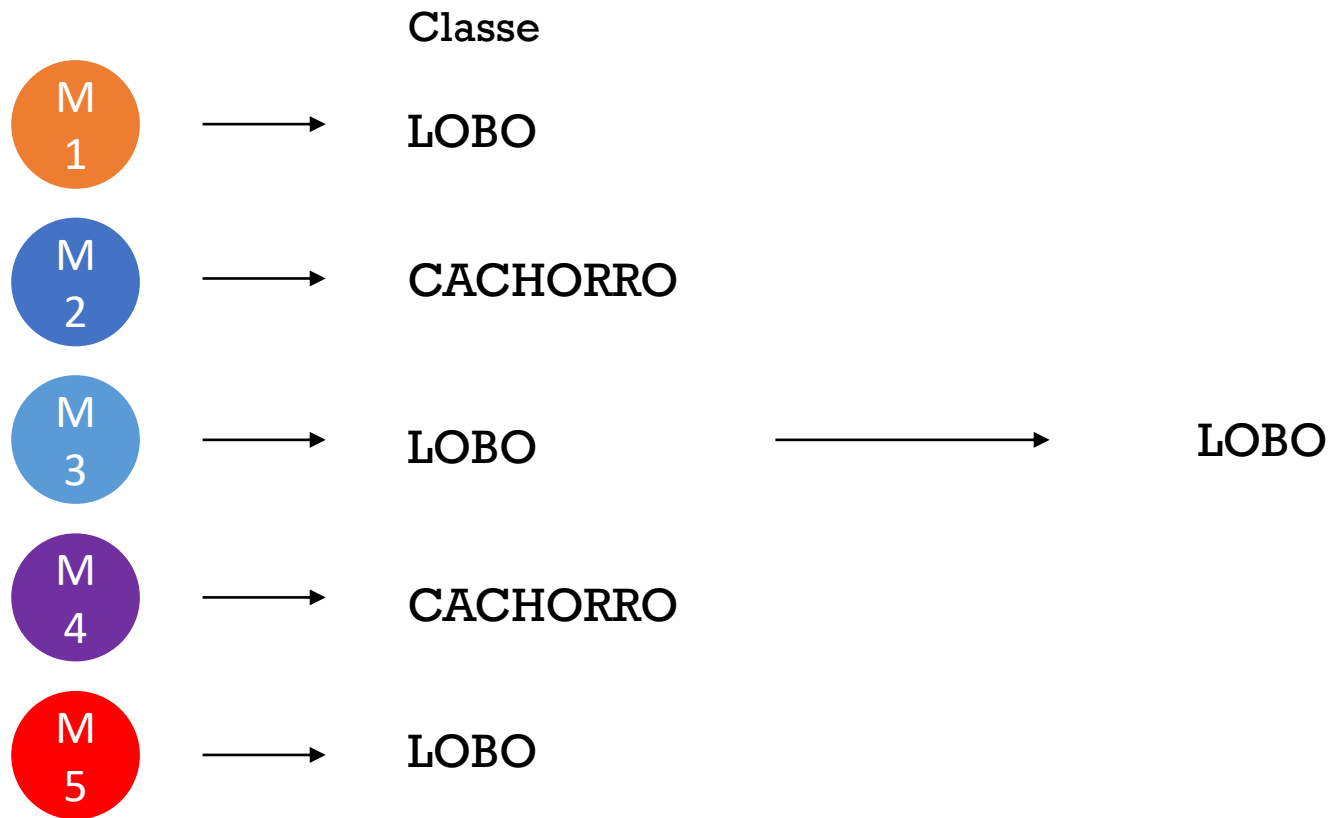
# Voting



Cachorro ou lobo?

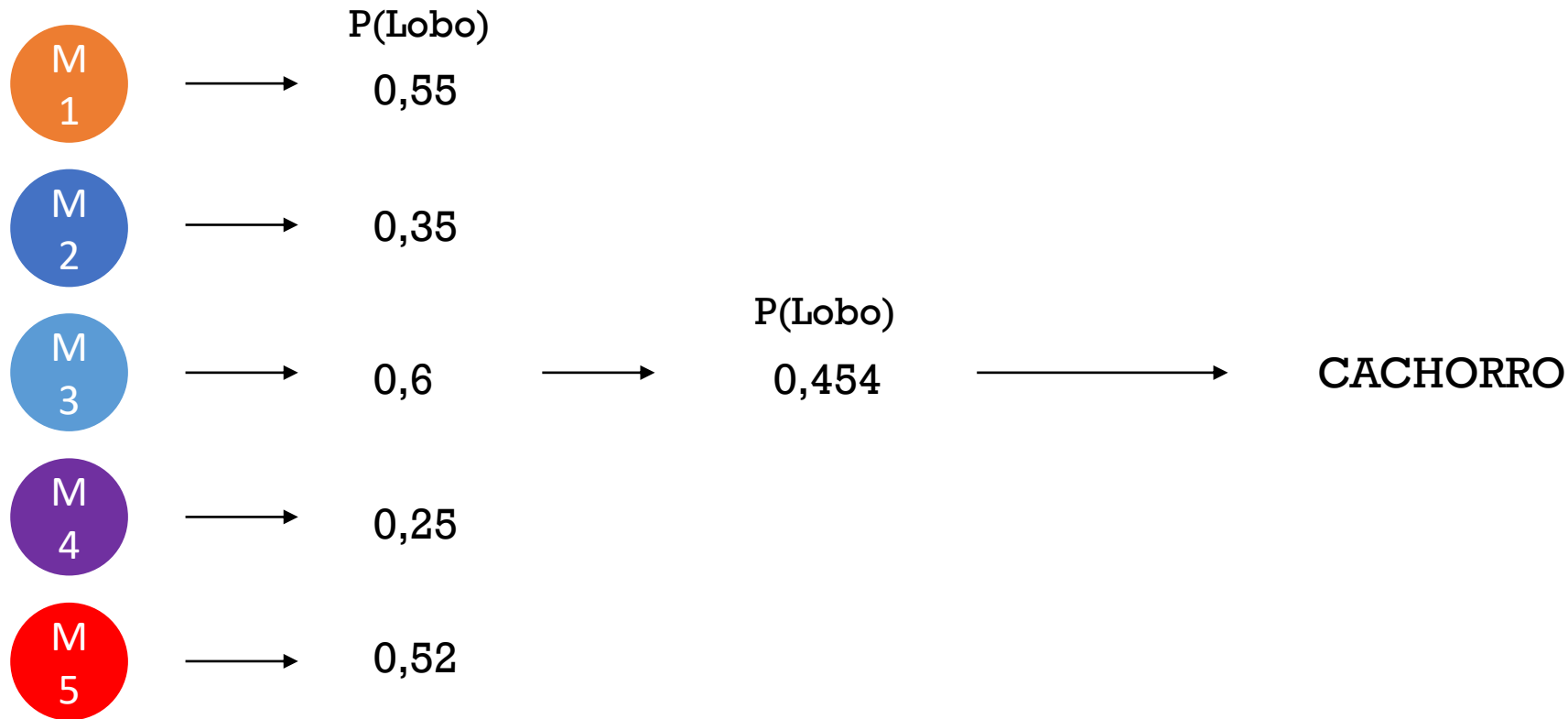
## Cachorro ou lobo?

# Max Voting



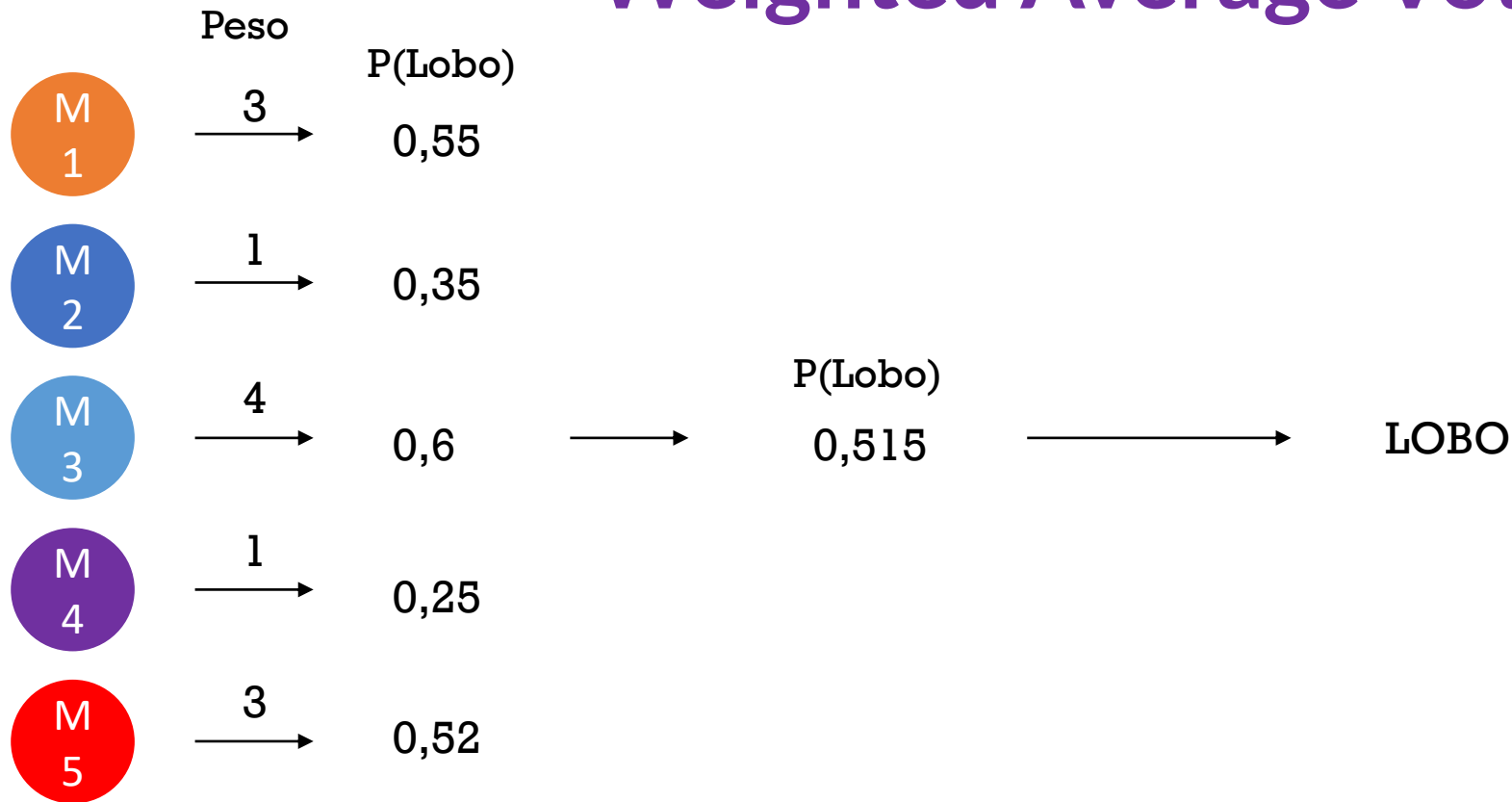
Cachorro ou lobo?

## Average Voting



Cachorro ou lobo?

# Weighted Average Voting

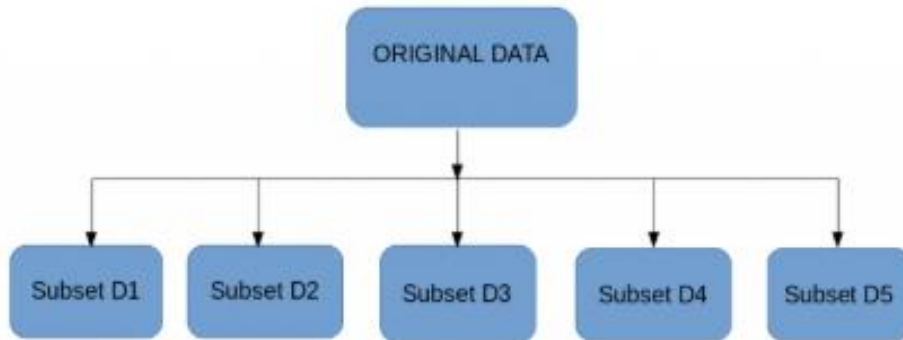


# Bagging

(Bootstrap aggregating)

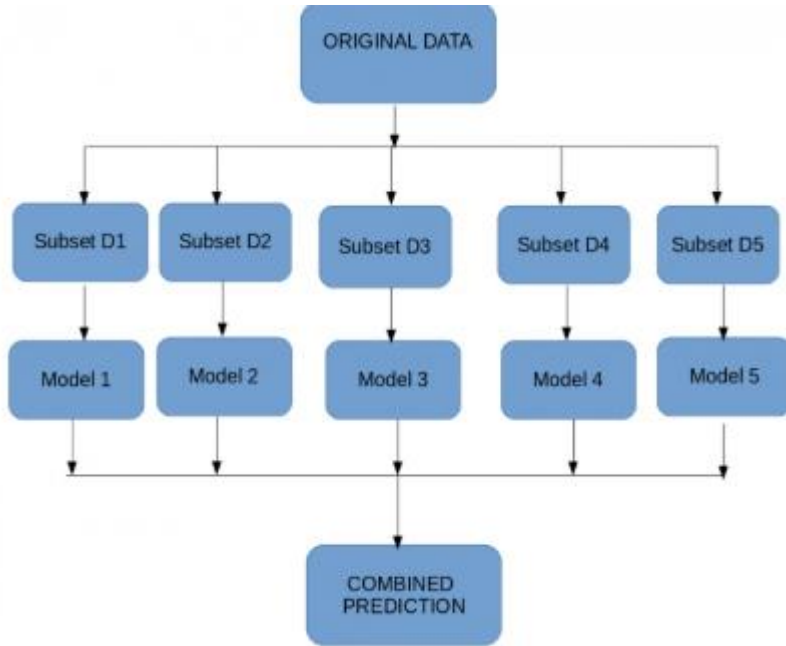
# Bagging

Conjunto de dados é dividido em subsets de mesmo tamanho usando uma amostragem randômica com reposição (Bootstrapping).



# Bagging

Cada subset gera um modelo e no final se obtém o resultado por votação.

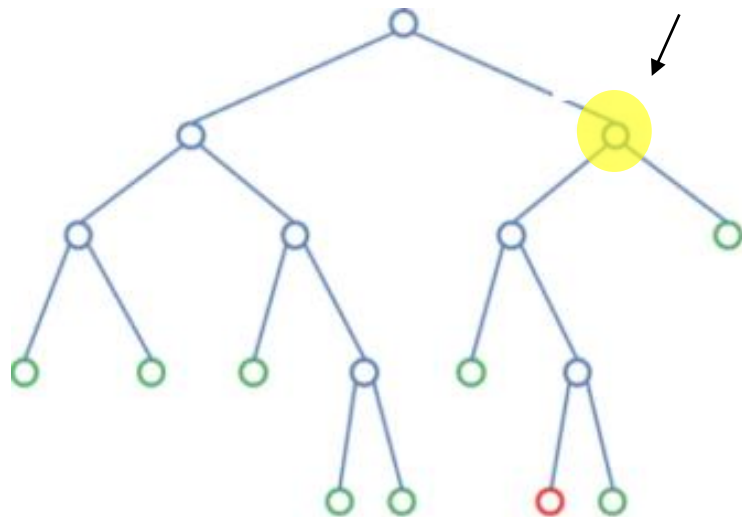




# Random Forests

# Random Forests

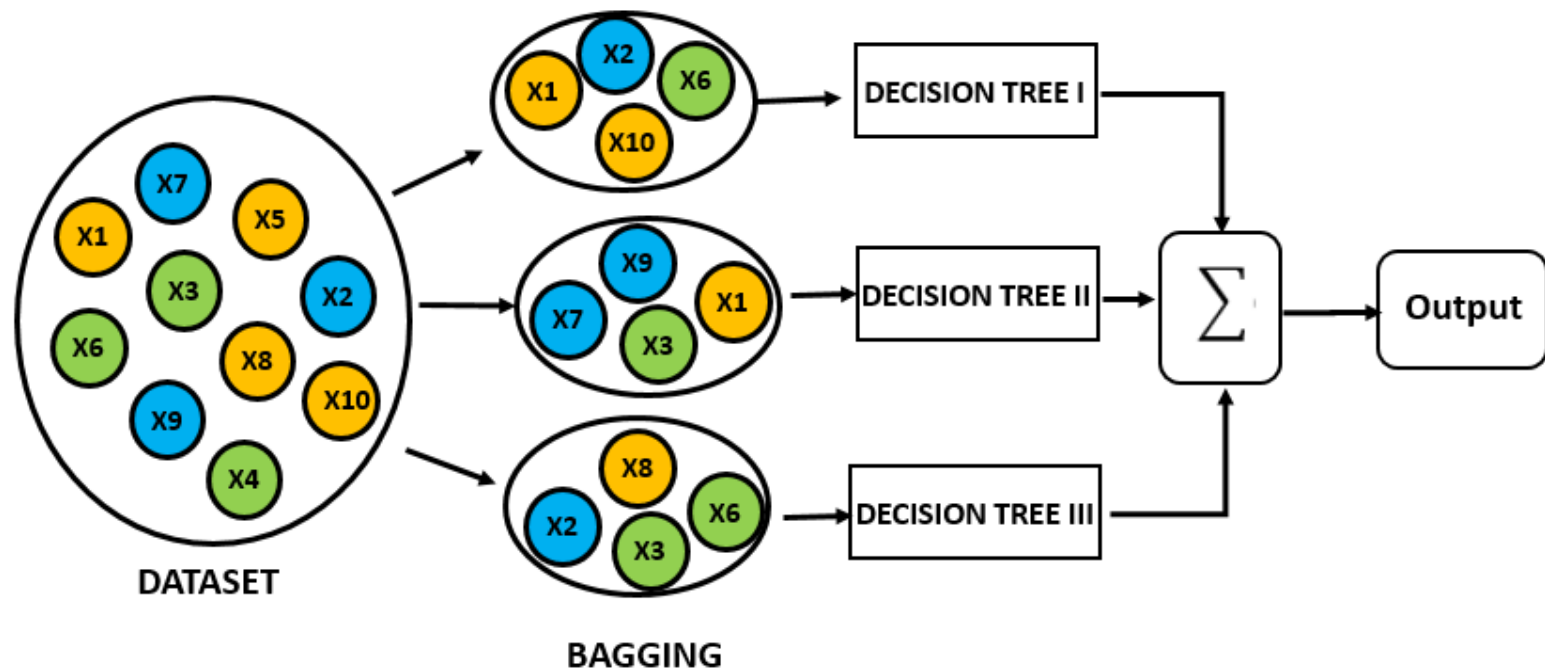
Feature bagging



Modelo que utiliza conceitos de Bagging, sendo cada subset gerado é treinado por uma árvore de decisão.

Além disso, faz “feature bagging” na hora de dividir cada nó, selecionado somente um subset das features.

# Random Forests



# Boosting

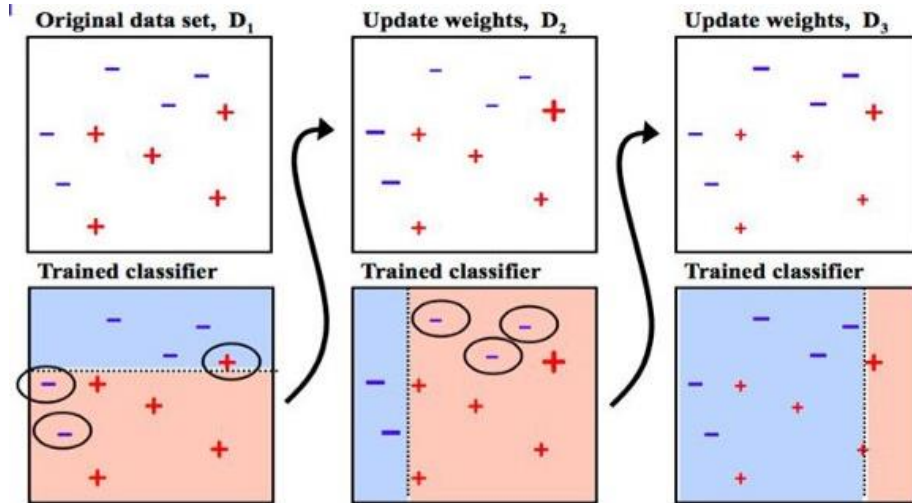
# Boosting

Ao invés de serem gerados em paralelo, modelos são gerados sequencialmente de acordo com os resultados dos modelos anteriores.

- Adaptive Boosting (AdaBoost)
- Gradient Boosting

# AdaBoosting

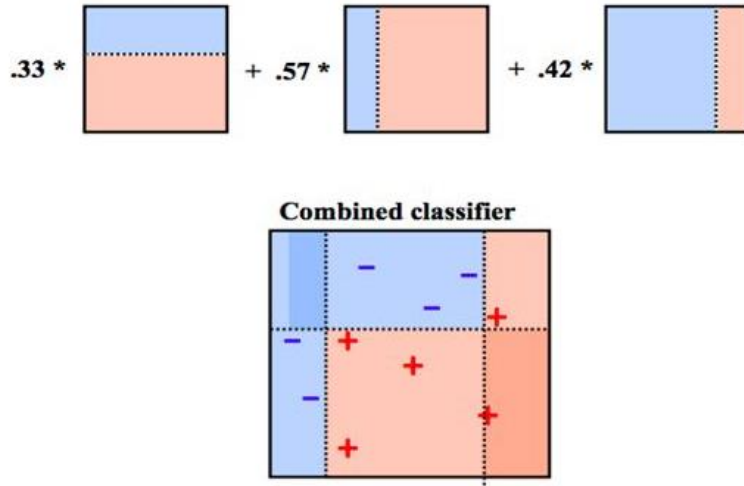
# AdaBoosting



Algoritmo que se baseia em dar pesos para as amostras, que são atualizados a cada etapa de acordo com os resultados dos modelos.

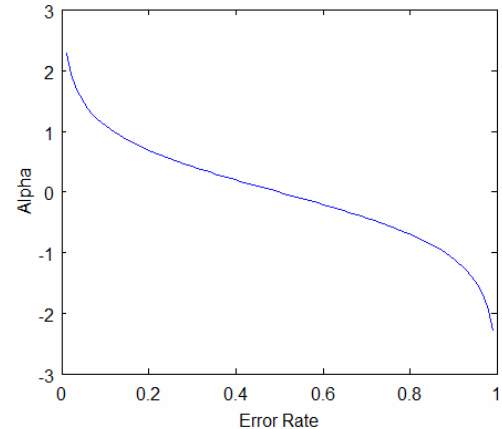
Podem ser utilizados diversos modelos, mas geralmente utilizam-se árvores de decisão.

# AdaBoosting



No final, os modelos são combinados usando pesos de acordo com o erro de cada modelo.

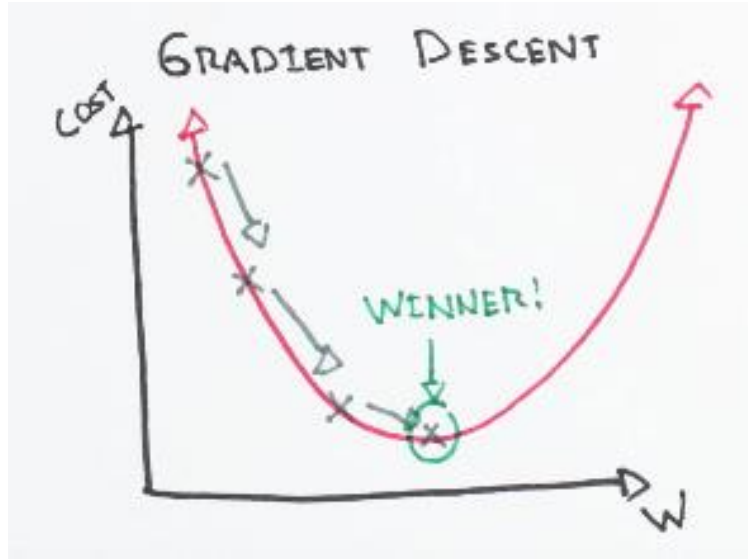
$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)$$





# Gradient Boosting

# Gradient Boosting



Método que utiliza ideias semelhante ao gradiente descente (Gradient Descent) para minimização de uma função de erro.

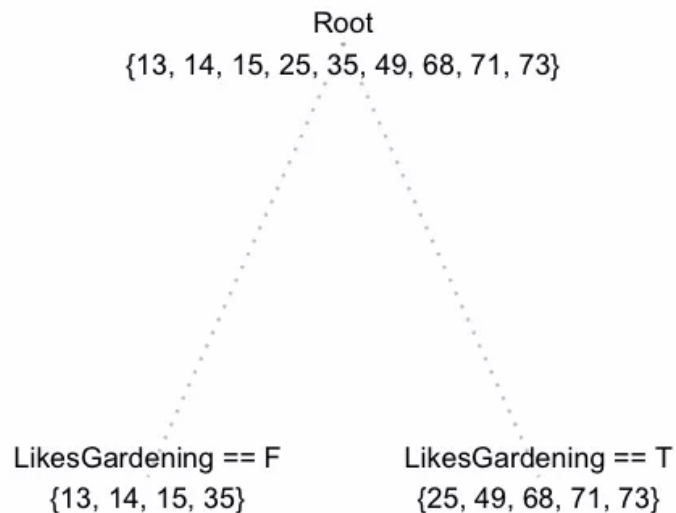
Também pode ser implementado com diversas funções, mas geralmente se utilizam árvores de decisão.

# Gradient Boosting

Intuição: Queremos prever a idade de uma pessoa baseado no fato dela gostar de jardinagem, videogames ou chapéus:

PersonID	Age	LikesGardening	PlaysVideoGames	LikesHats
1	13	FALSE	TRUE	TRUE
2	14	FALSE	TRUE	FALSE
3	15	FALSE	TRUE	FALSE
4	25	TRUE	TRUE	TRUE
5	35	FALSE	TRUE	TRUE
6	49	TRUE	FALSE	FALSE
7	68	TRUE	TRUE	TRUE
8	71	TRUE	FALSE	FALSE
9	73	TRUE	FALSE	TRUE

# Gradient Boosting



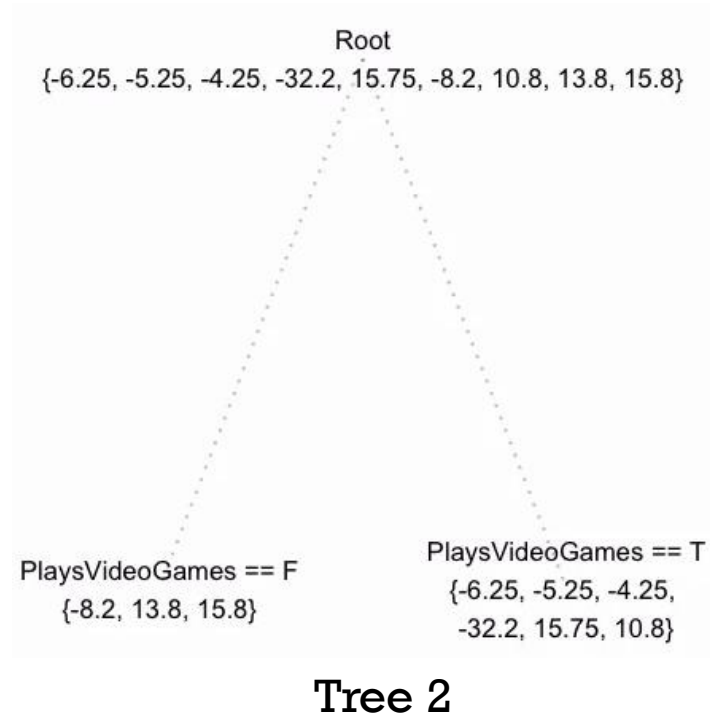
**Tree 1**



ID	Age	Prediction 1	Residual 1
1	13	19,25	-6,25
2	14	19,25	-5,25
3	15	19,25	-4,25
4	25	57,2	-32,2
5	35	19,25	15,75
6	49	57,2	-8,2
7	68	57,2	10,8
8	71	57,2	13,8
9	73	57,2	15,8

# Gradient Boosting

Residual 1
-6,25
-5,25
-4,25
-32,2
15,75
-8,2
10,8
13,8
15,8



# Gradient Boosting

ID	Age	Prediction 1	Residual 1	Prediction 2	Combined Prediction	Residual
1	13	19,25	-6,25	-3,567	15,68	2,683
2	14	19,25	-5,25	-3,567	15,68	1,683
3	15	19,25	-4,25	-3,567	15,68	0,6833
4	25	57,2	-32,2	-3,567	53,63	28,63
5	35	19,25	15,75	-3,567	15,68	-19,32
6	49	57,2	-8,2	7,133	64,33	15,33
7	68	57,2	10,8	-3,567	53,63	-14,37
8	71	57,2	13,8	7,133	64,33	-6,667
9	73	57,2	15,8	7,133	64,33	-8,667

# Gradient Boosting

Esse algoritmo inicial pode ser estendido usando  $M$  modelos, gerando repetidas vezes novos modelos treinados nos resíduos do modelo anterior.

O algoritmo final é uma adaptação desse algoritmo inicial e pode ser explicado em 7 passos.

# Gradient Boosting

1. Iniciar modelo simples como a média dos valores

$$F_o(x) = \hat{y}$$

2. Definir função de perda (L) que depende do modelo inicial e dos valores de y

$$L = L(y, F_o(x))$$



# Gradient Boosting

3. Calcular pseudo-resíduos ( $r$ ) para o primeiro modelo usando a derivada do erro  $L$ .

$$r_o = - \frac{\delta L(y, F_o(x))}{\delta F_o(x)}$$

4. Treinar próximo modelo ( $h_0$ ) usando  $r$  como input do modelo.

# Gradient Boosting

5. Calcular peso ( $\gamma$ ) associado ao modelo, minimizando a função do erro:

$$\gamma_o = \operatorname{argmin}(L(y, F_o(x) + \gamma_o h_o(x)))$$

6. Gerar novo modelo, usando learning rate  $v$

$$F_1(x) = F_0(x) + v \gamma_o h_o(x)$$

7. Repetir M vezes

# Gradient Boosting

Além desses 7 passos, há duas modificações importantes para reduzir overfitting e melhor generalização do modelo:

- A cada novo modelo gerado é selecionado apenas um subset dos dados para treinar (similar ao Bagging)
- O mesmo é feito com as colunas, sendo selecionado apenas um subset a cada iteração.

# XGBoost

# XGBoost

Implementação lançada em 2014 como um projeto de pesquisa por Tianqi Chen.

Suporte para diversas linguagens (C++, Java, Python, R, Julia) e frameworks para dados distribuídos (Apache Hadoop, Apache Spark). Pode ser treinado usando CPU ou GPU.

Implementa diversas otimizações que tornam o treinamento mais rápido e obtêm mais acurácia que a implementação base de Gradient Boosted Trees.

Popular no Kaggle, em especial 2016-2017 (Higgs Boson competition).

# LightGBM

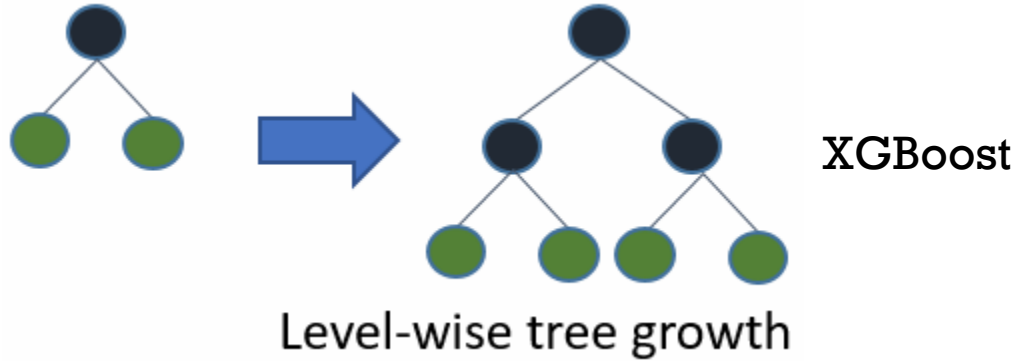
# LightGBM

Lançado em Janeiro de 2017 pela Microsoft como uma melhoria do XGBoost, especialmente na velocidade de treino.

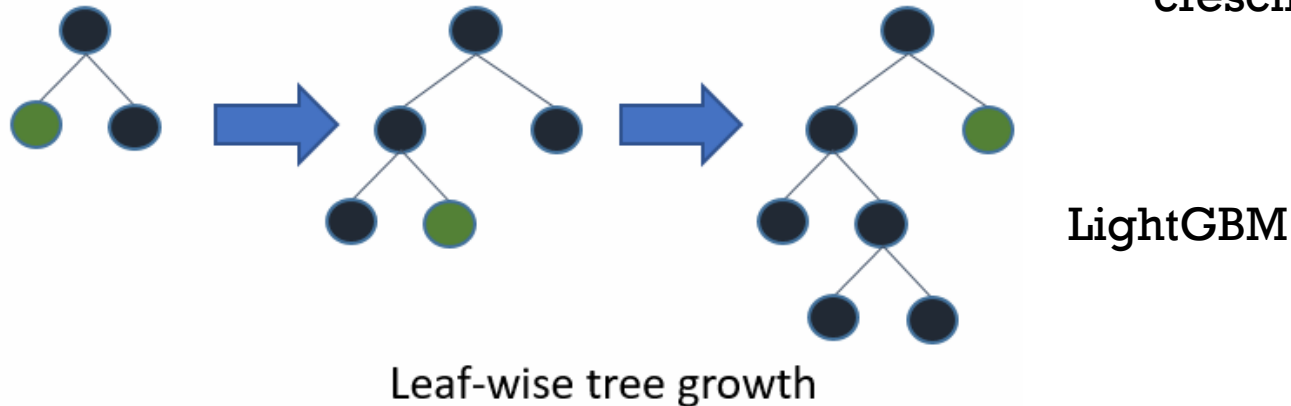
Ao contrário do XGBoost, tem algoritmo próprio para lidar com valores categóricos, sem precisar usar one-hot encoding (no entanto, categorias devem ser convertidas para números).

Método mais utilizado hoje no Kaggle devido a sua velocidade de treinamento.

# LightGBM



Uma das diferenças em relação ao XGBoost (além do tratamento das features categóricas) é a forma de crescimento da árvore.





**CatBoost**

# CatBoost

Lançado em Abril de 2017 pela Yandex (maior empresa de tecnologia e motor de busca da Rússia).

Também tem implementação própria para lidar com variáveis categóricas mas sem necessidade de converter categorias para números.

Em média, velocidade entre XGBoost e LightGBM utilizando CPU e mais rápido que LightGBM utilizando GPU.

Implementação menos utilizada que as outras duas.

# Hiperparâmetros

As três implementações possuem diversos hiperparâmetros, mas alguns dos principais e comuns aos três são:

- N\_estimators
- Learning rate
- Max\_depth
- Subsample
- Colsample
- Regularization alpha e lambda

# Implementação

# Implementação

Exemplo simples para demonstrar aplicação dos três modelos.

Todos os modelos usaram os hiperparametros default e 200 árvores.

O dataset é baseado no censo americano de 1994, cujo objetivo é prever renda ( $<50k$  ou  $>50k$ ) com base em dados demográficos:

Dataset original e código da implementação:

[archive.ics.uci.edu/ml/datasets/Census+Income](http://archive.ics.uci.edu/ml/datasets/Census+Income)

[github.com/luiznonenmacher/classical-machine-learning/tree/master/finding\\_donors\\_boosting](https://github.com/luiznonenmacher/classical-machine-learning/tree/master/finding_donors_boosting)

# Implementação

	age	workclass	education_level	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	Bachelors	13.0	Never-married	Adm-clerical	Not-in-family	White	Male	2174.0	0.0	40.0	United-States	<=50K
1	50	Self-emp-not-inc	Bachelors	13.0	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.0	0.0	13.0	United-States	<=50K
2	38	Private	HS-grad	9.0	Divorced	Handlers-cleaners	Not-in-family	White	Male	0.0	0.0	40.0	United-States	<=50K
3	53	Private	11th	7.0	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0.0	0.0	40.0	United-States	<=50K
4	28	Private	Bachelors	13.0	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0.0	0.0	40.0	Cuba	<=50K
5	37	Private	Masters	14.0	Married-civ-spouse	Exec-managerial	Wife	White	Female	0.0	0.0	40.0	United-States	<=50K
6	49	Private	9th	5.0	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0.0	0.0	16.0	Jamaica	<=50K
7	52	Self-emp-not-inc	HS-grad	9.0	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.0	0.0	45.0	United-States	>50K
8	31	Private	Masters	14.0	Never-married	Prof-specialty	Not-in-family	White	Female	14084.0	0.0	50.0	United-States	>50K
9	42	Private	Bachelors	13.0	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178.0	0.0	40.0	United-States	>50K

# Implementação

	age	education- num	capital- gain	capital- loss	hours- per- week	workclass_ Federal- gov	workclass_ Local-gov	workclass_ Private	workclass_ Self-emp- inc	workclass_ Self-emp- not-inc	...	native- country_ Portugal	native- country_ Puerto- Rico	native- country_ Scotland	c
13181	0.410959	0.600000	0.0	0.000000	0.500000	0	0	1	0	0	...	0	0	0	
10342	0.438356	0.533333	0.0	0.000000	0.397959	0	0	1	0	0	...	0	0	0	
20881	0.054795	0.666667	0.0	0.000000	0.357143	0	0	1	0	0	...	0	0	0	
24972	0.301370	0.866667	0.0	0.905759	0.448980	0	1	0	0	0	...	0	0	0	
43867	0.246575	0.600000	0.0	0.000000	0.500000	0	0	1	0	0	...	0	0	0	

36177 linhas no conjunto de treino e 9045 no de teste.  
103 colunas.

# Implementação

## XGBoost

```
import xgboost as xgb

start = time()
xg = xgb.XGBClassifier(n_estimators=200, random_state=42, n_jobs=4)
xg.fit(X_train, y_train)
end = time()

xgb_training_time = end - start

train_predictions = xg.predict(X_train)
test_predictions = xg.predict(X_test)

xgb_train_auc = accuracy_score(train_predictions, y_train)
xgb_test_auc = accuracy_score(test_predictions, y_test)

print('Training time: {}, train_acc: {}, test_acc = {}'.format(round(xgb_training_time,2),
                                                                round(xgb_train_auc,4),
                                                                round(xgb_test_auc,4)))
```

Training time: 3.87, train\_acc: 0.8715, test\_acc = 0.8672



# Implementação

## LightGBM

```
import lightgbm as lgb

start = time()
lg = lgb.LGBMClassifier(n_estimators=200, random_state=42, n_jobs=4)
lg.fit(X_train, y_train)
end = time()
lgb_training_time = end - start

train_predictions = lg.predict(X_train)
test_predictions = lg.predict(X_test)

lgb_train_auc = accuracy_score(train_predictions, y_train)
lgb_test_auc = accuracy_score(test_predictions, y_test)

print('Training time: {}, train_acc: {}, test_acc = {}'.format(round(lgb_training_time,2),
                                                                round(lgb_train_auc,4),
                                                                round(lgb_test_auc,4)))
```

Training time: 0.46, train\_acc: 0.8862, test\_acc = 0.8704

# Implementação

## CatBoost

```
import catboost as ctb

start = time()
cb = ctb.CatBoostClassifier(n_estimators=200, verbose=False, random_state=42)
cb.fit(X_train, y_train)
end = time()
cb_training_time = end - start

train_predictions = cb.predict(X_train)
test_predictions = cb.predict(X_test)

cb_train_auc = accuracy_score(train_predictions, y_train)
cb_test_auc = accuracy_score(test_predictions, y_test)

print('Training time: {}, train_acc: {}, test_acc = {}'.format(round(cb_training_time,2),
                                                                round(cb_train_auc,4),
                                                                round(cb_test_auc,4)))
```

Training time: 9.21, train\_acc: 0.8769, test\_acc = 0.871

# Implementação

Resultados:

Modelo	Training time (s)	Test Accuracy
XGBoost	3,87	0,8672
LightGBM	<b>0,46</b>	0,8704
CatBoost	9,21	<b>0,8710</b>

# Contatos



[ljuniornone@gmail.com](mailto:ljuniornone@gmail.com)



[github.com/luiznonenmacher](https://github.com/luiznonenmacher)



[www.linkedin.com/in/luiz-nonenmacher](https://www.linkedin.com/in/luiz-nonenmacher)

**Muito obrigado pela atenção!**