Decision Trees com Apache Spark

Predição de cobertura florestal usando Decision Trees com Apache Spark

Alunos:

João Antonio Ferreira *joao.parana@gmail.com*

Rodrigo Tavares de Souza *rtavaresrj87@gmail.com*

Professor:

Eduardo Ogasawara

http://eic.cefet-rj.br/ppcic

Motivadores e Objetivo

- Florestas
  - Diversidade de fauna e flora
  - Ecossistema é orgânico
  - Conservação dos rios que fornecem água potável
  - Sequestro de Carbono
  - Necessidade de preservação
  - Desmatamento leva a degradação do solo

- Nosso Objetivo:
  - Predição de cobertura florestal usando Aprendizado de Máquina

Problema & proposta

Dado um conjunto de informações topográficas prever o tipo de vegetação que melhor se adapta as condições dadas

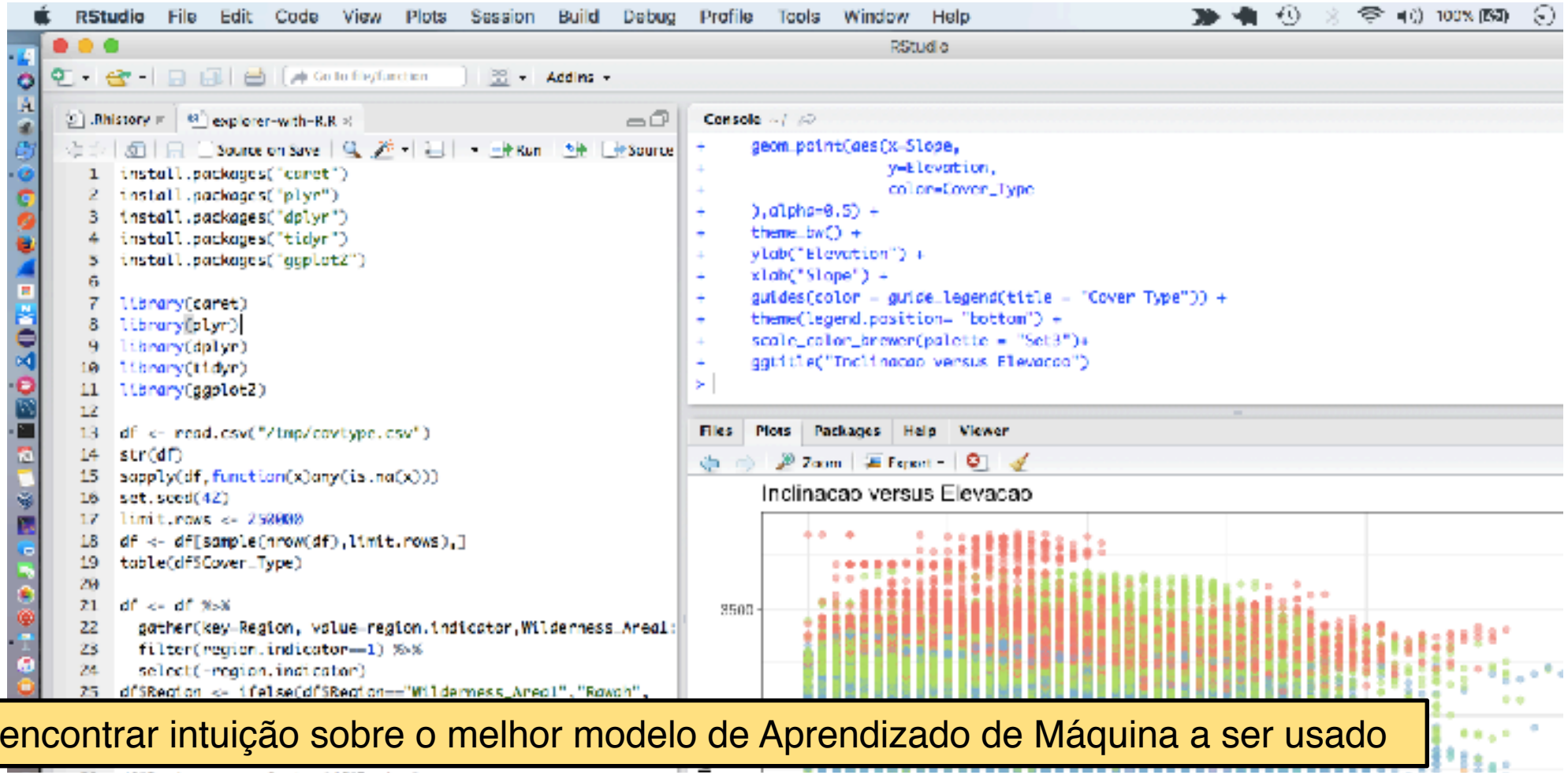
- Problema de Classificação
- Aprendizado de máquina - Supervisionando
- Qual modelo ? Arvore de Decisão e *RandomForest*

Metodologia

Análise Exploratória usando linguagem R

Fazer análise do Schema e dos dados

Dataset



Permite encontrar intuição sobre o melhor modelo de Aprendizado de Máquina a ser usado

Metodologia

Dataset

Análise Exploratória usando R

Distribuição por Região

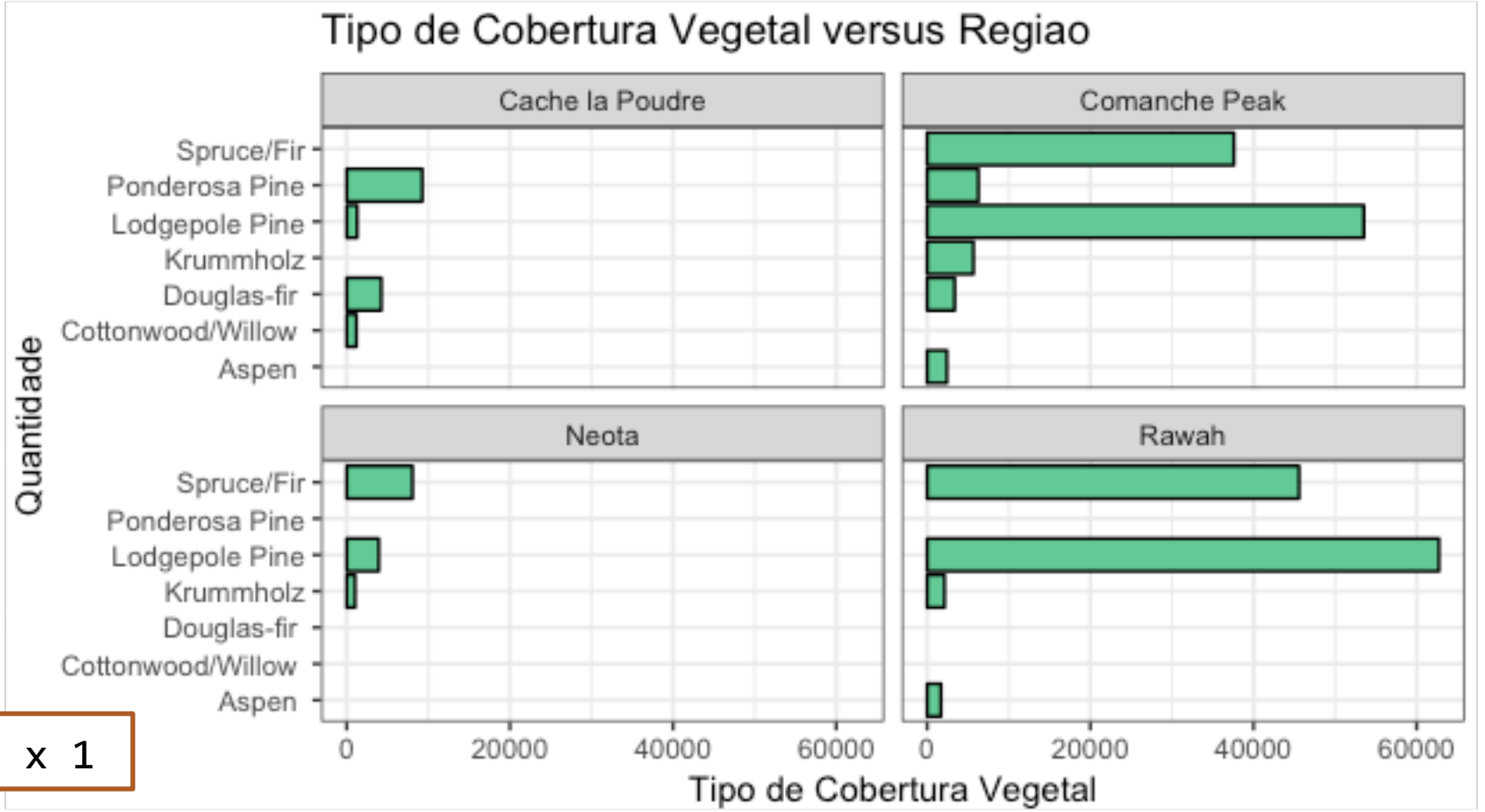
Análise do Schema

e dos dados

```
set.seed(42)
limit.rows <- 250000
df <- df[sample(nrow(df),limit.rows),]
table(df$Cover_Type)
```

Desbalanceamento das classes : 120 x 1

```
##
##     1     2     3     4     5     6     7
## 91244 121478 15530  1177  4119  7608  8844
```



Metodologia

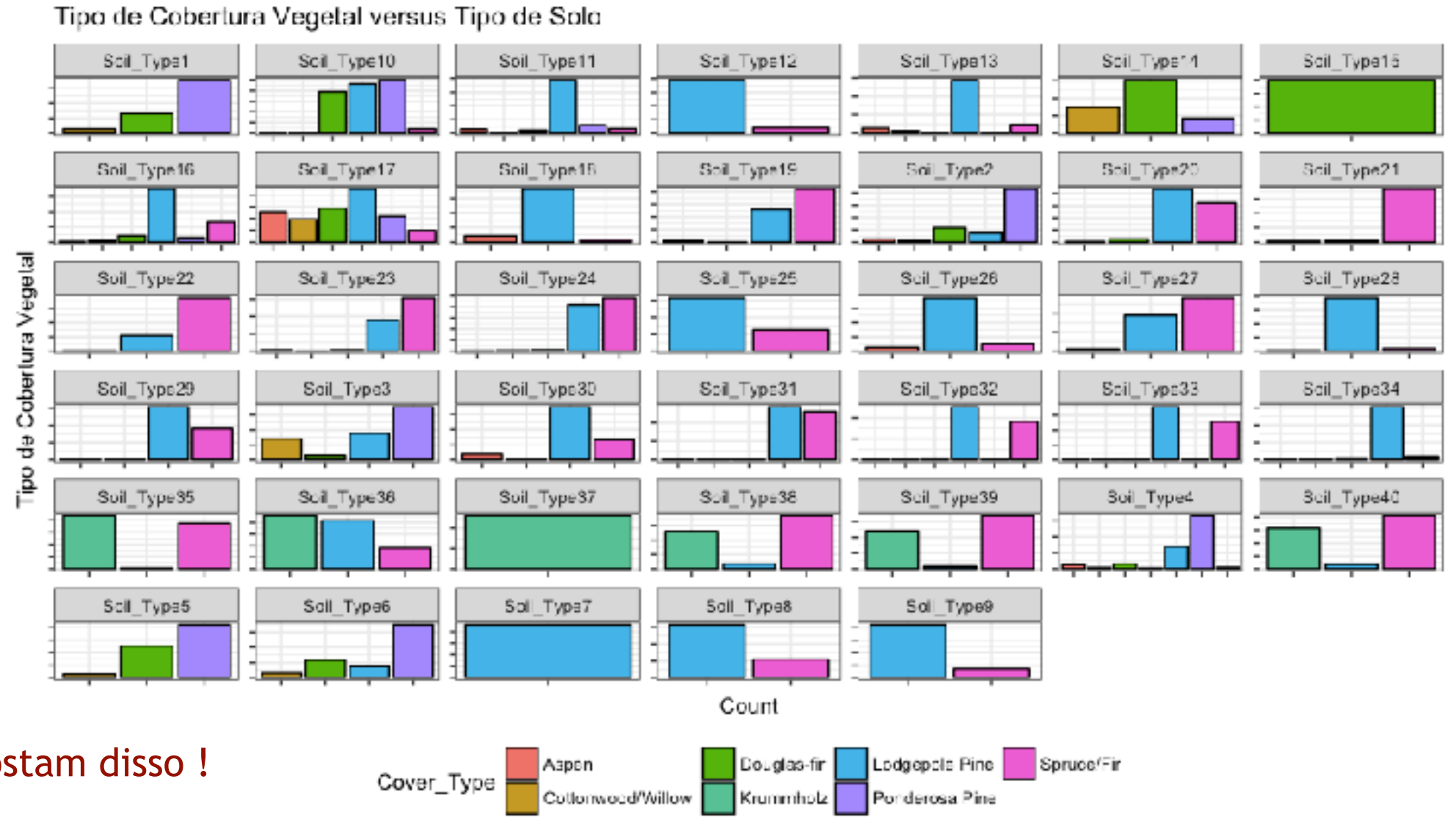
Dataset

Análise do Schema

Análise Exploratória usando linguagem R

Alguns tipos de solo não possuem certos tipos de cobertura vegetal !

Arvores de decisão gostam disso !



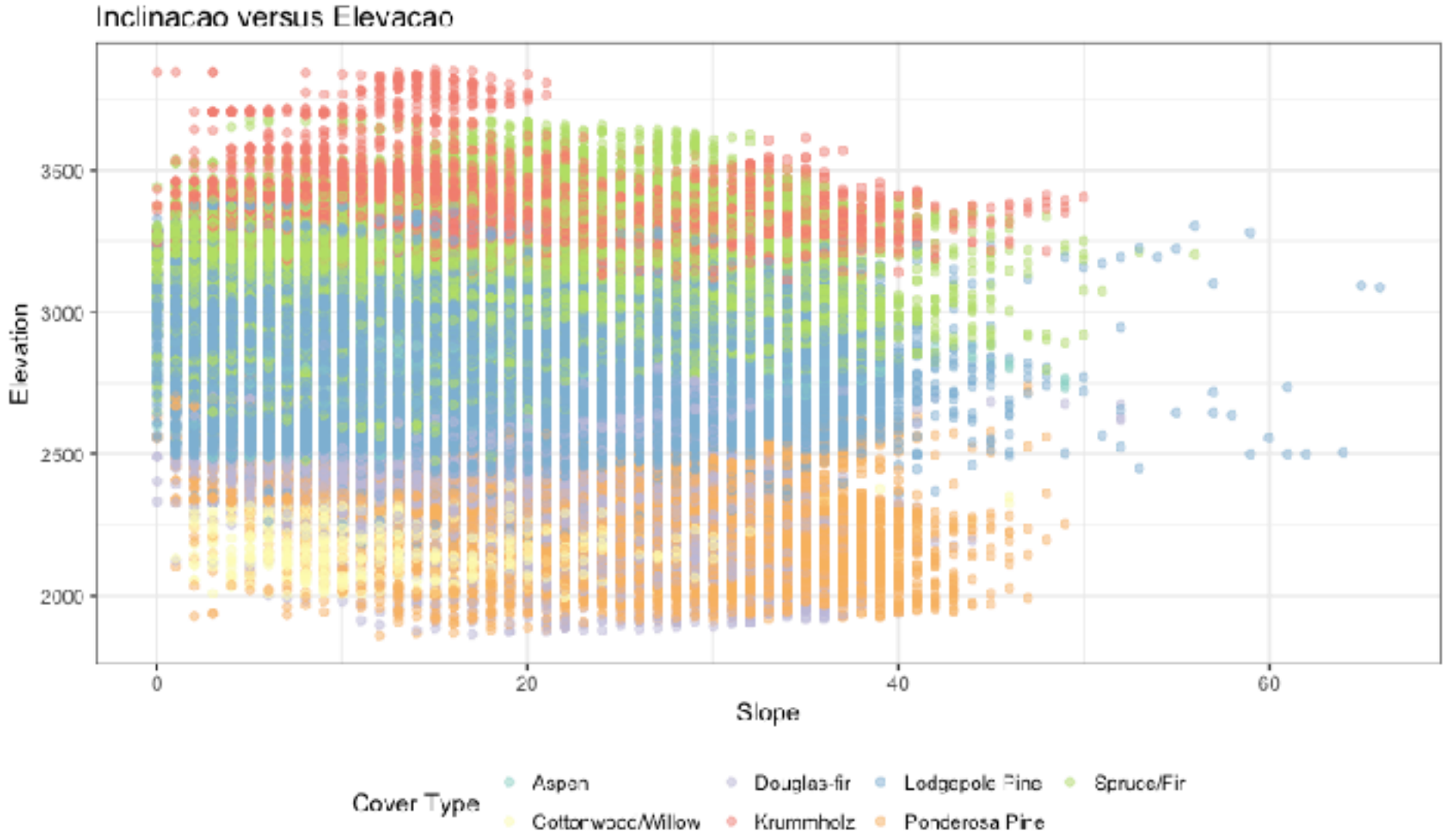
Metodologia

Dataset

Análise do Schema

Análise Exploratória usando linguagem R

Elevação é mais importante que inclinação !



Metodologia

- Datasets com dezenas de Features
  - *one-hot encoded* para dados categóricos
- Datasets muito grandes
- Necessidade de processamento eficiente

Trata-se de um problema de BigData

Usaremos Apache Spark

Spark - Características

- Ecossistema Spark com linguagem Scala
- Abstração de infraestrutura complexa
- Resiliência - recupera-se de falhas
- MLLib - biblioteca para Aprendizado de Maquina
- SparkSQL e API Dataset
- Diversos algoritmos
  - Logistic regression, naive Bayes, generalized linear regression, survival regression, decision trees, random forests, gradient-boosted trees, K-means, frequent itemsets, association rules, sequential pattern mining

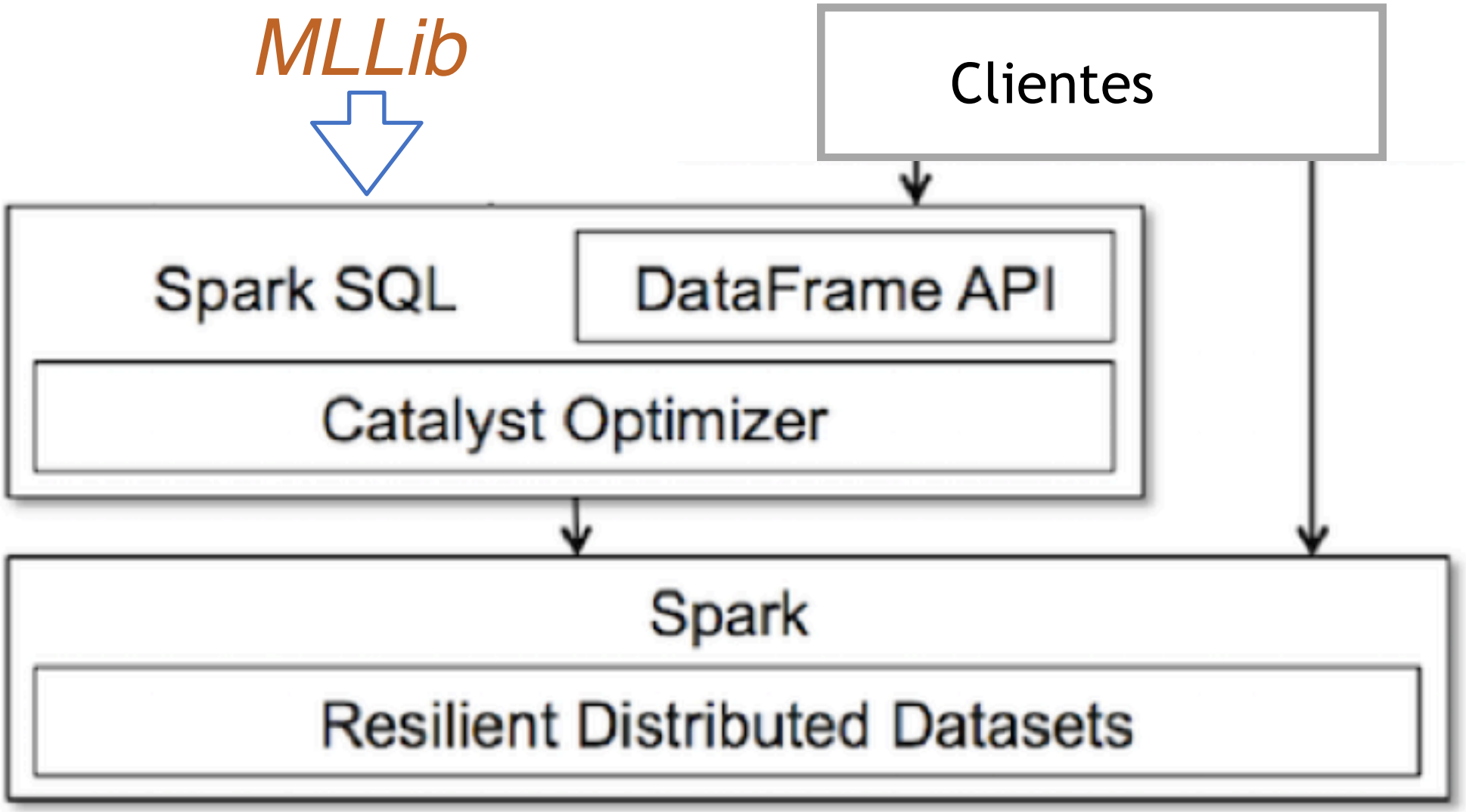
Spark - Características

- Utilitários
  - Feature transformations (standardization, normalization, hashing), model evaluation and hyper-parameter tuning, ML persistence, distributed linear algebra (SVD, PCA), statistics: summary statistics, hypothesis testing

- Função vai aos Dados. É outro Paradigma !
- Programação funcional com operadores *map*, *reduce*, *filter*, etc
- MLLib implementa *DecisionTree* e *RandomForest*

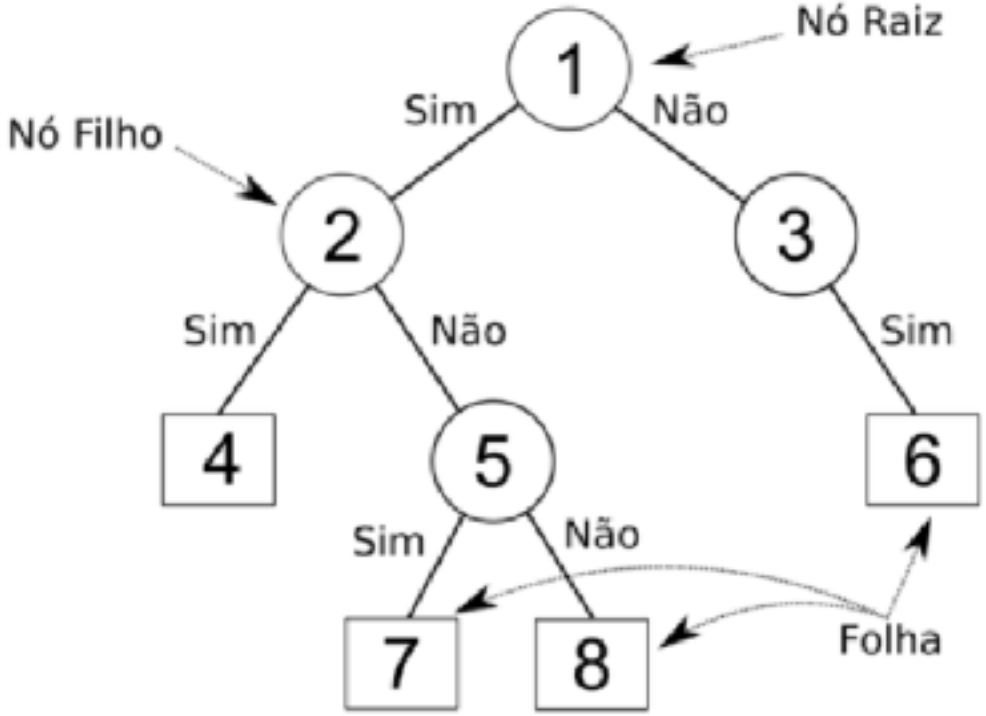
Spark - Arquitetura

Arquitetura



Arvore de Decisão

- Classificação (predizer valores discretos ou classes)
- Regressão (predizer valores contínuos)
- Estimativa de probabilidade
- Agrupamentos (clustering)

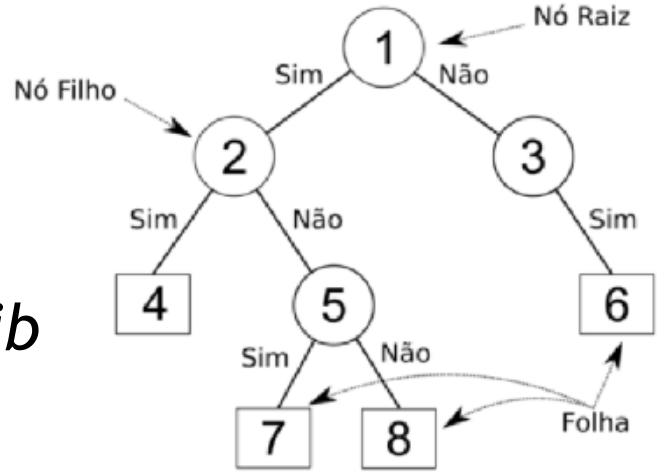


- Resultado de fácil interpretação
- Dados não precisam ser normalizados
- São robustas à valores extremos (outliers)
- São paralelizáveis em ambiente de HPC
- Consomem dados de tipos diferentes
- Conjuntos de árvores : *ensemble method is Random Forests*

Nó raiz e nós filhos são predicados a serem avaliados
Folhas são alvos da classificação (cobertura vegetal)

Arvore de Decisão

- A chave é a escolha da *feature* usada para dividir a arvore
  - tipo de solo, região, elevação, . . . ?
  - mais puras (*impurity* baixa) vem primeiro
- Critério é escolher pelo melhor ganho de informação
- *Impurity*
  - G - *Gini* e E - *Entropy são implementados no Spark MLLib*



$$I_G(p) = 1 - \sum_{i=1}^{N} p_i^2$$

$$I_E(p) = \sum_{i=1}^{N} p_i \log\left(\frac{1}{p}\right) = -\sum_{i=1}^{N} p_i \log\left(p_i\right)$$

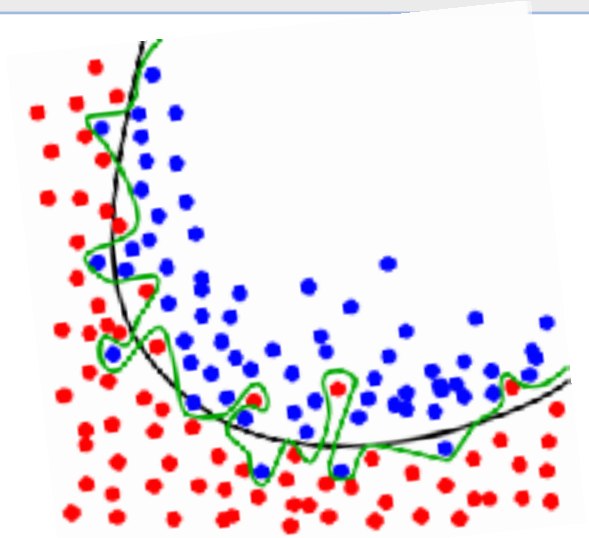
p_i proporção de exemplos da classe ***i***

N quantidade de classes

Arvore de Decisão

- Evitar o Overfitting
  - Limitar o número de níveis nas Arvores de Decisão tree.
  - Usar Random Forests
  - Verificar boa acurácia no cross-validation

- Melhorar a acurácia
  - Usar Entropy em vez de Gini
  - Configurar um numero maior de bins (Ex.: 300)
  - Escolher 30 como profundidade máxima da árvore

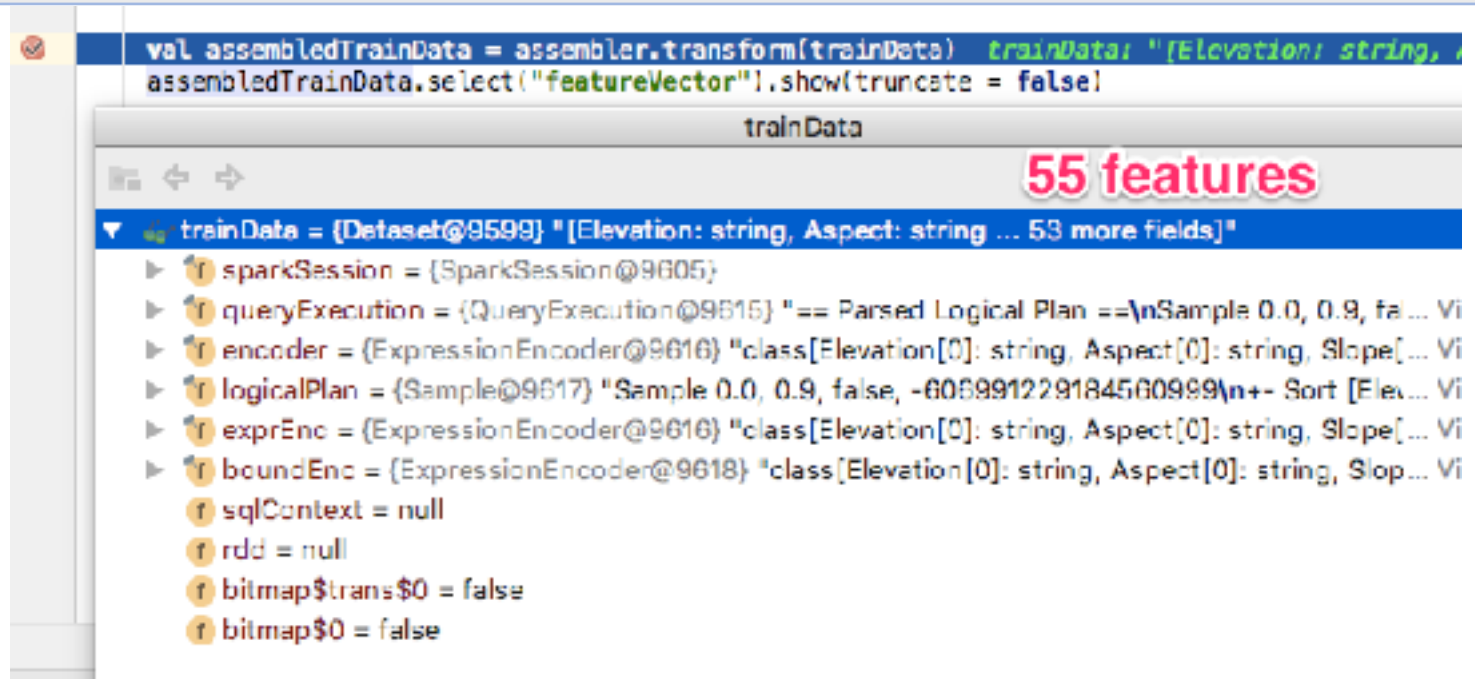


Sobre-ajuste ou Overfitting

Quanto melhor a Acurácia maior é o Custo computacional quase sempre !

Avaliação Experimental

- RandomForest
  - 7 - quantidade de targets
    - tipos de cobertura vegetal
  - 55 features
  - Ajuste da estrutura dos dados
  - Possibilidade de DEBUG




```
val forest = RandomForest.trainClassifier(
  trainData, 7, Map(10 -> 4, 11 -> 40), 20,
  "auto", "entropy", 30, 300)
```

Avaliação Experimental


```
val input = "2709,125,28,67,23,3224,253,207,61,6094,0,29"
val vector = Vectors.dense(input.split(',').map(_.toDouble))
forest.predict(vector) ❶
```

- Dado um Array, podemos prever a cobertura vegetal

Avaliação Experimental

baixo paralelismo x custo de sincronização/memória x custo de rede

$$elapsed\_time \rightarrow T_1, T_2, T_3, T_4, T_5, T_6, \ldots T_p$$ Tempo decorrido

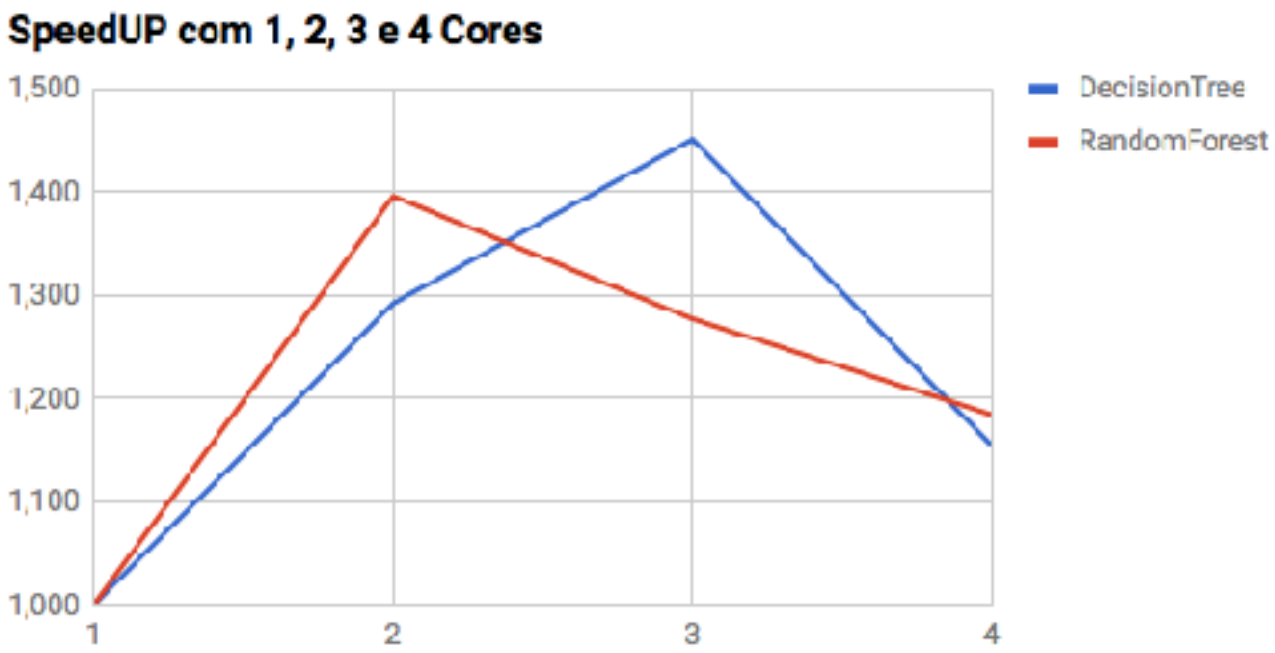
$$speed\_up \rightarrow S_{up} = \frac{T_1}{T_p}$$ Speed-up

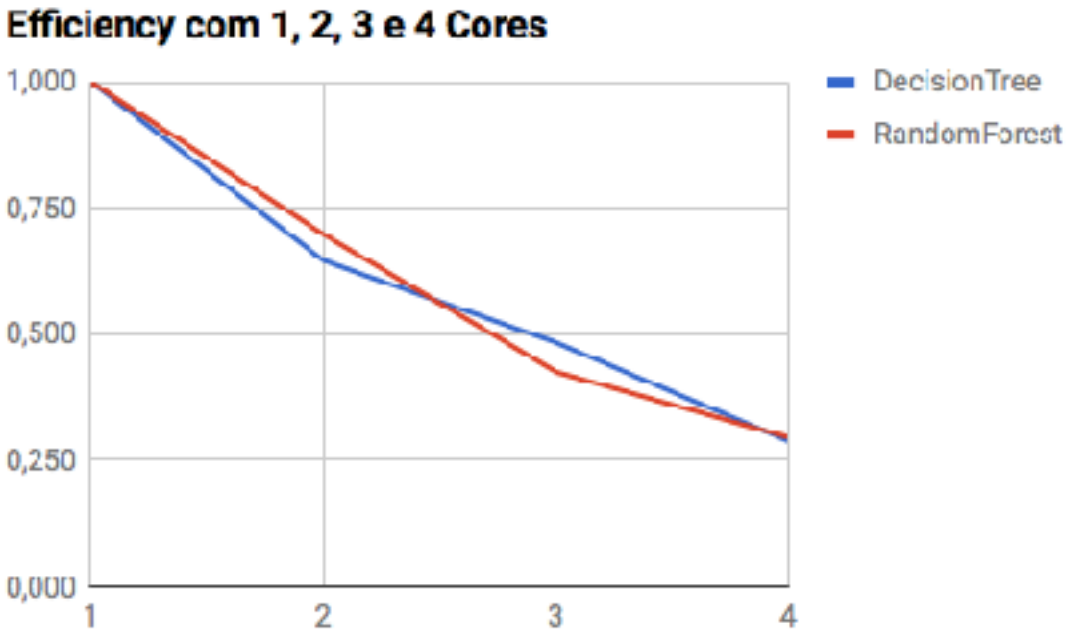
$$\xi = \frac{S_{up}}{p}$$ Eficiência

$$serial\_fraction \rightarrow f = \frac{\frac{1}{S_{up}(p)} - \frac{1}{p}}{1 - \frac{1}{p}}$$ Fração Serial

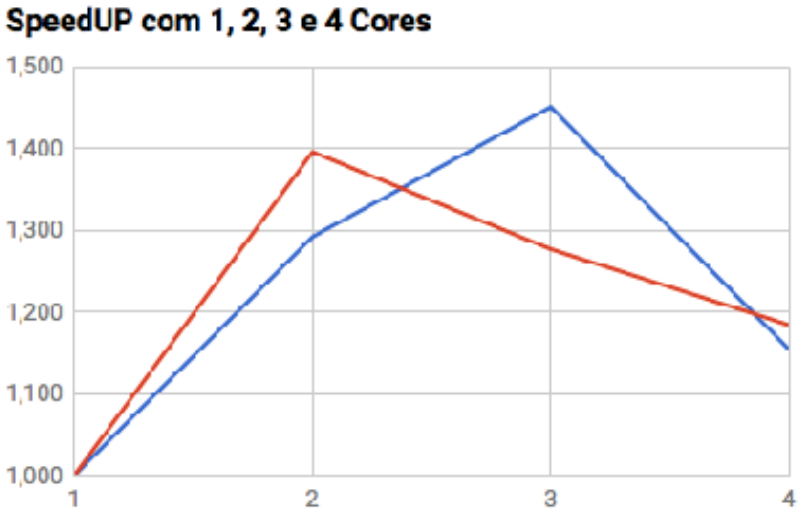
Avaliação Experimental

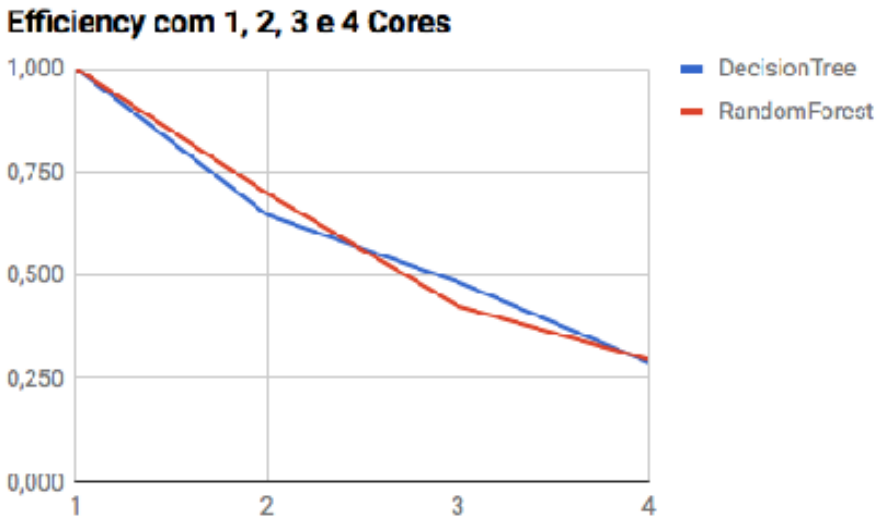
	SpeedUP		Efficiency			
Cores	DecisionTree	RandomForest	DecisionTree	RandomForest	Ellapsed DT	Ellapsed DF
1	1,000	1,000	1,000	1,000	56.061	833.023
2	1,292	1,396	0,646	0,698	43.396	596.917
3	1,451	1,277	0,484	0,426	38.641	652.566
4	1,154	1,184	0,289	0,296	48.577	703.708





Discussão





Qual a razão de piorar para 4 cores ?
DEBUG dos planos de execução ajudaria ?
Avaliar também a Fração Serial ?
Investigar a Implementação do Spark ?
Necessidade de testes em Cluster com até 16 cores.

- Ciclo de testes mais demorado

Dúvidas & Sugestões

Predição de cobertura florestal usando Decision Trees com Apache Spark

Alunos:

João Antonio Ferreira *joao.parana@gmail.com*

Rodrigo Tavares de Souza *rtavaresrj87@gmail.com*

Professor:

Eduardo Ogasawara

Agradecimentos