

# Projeto 3 - Florestas Randômicas

## Introdução à Inteligência Artificial

### Professor DÍbio Leandro Borges

Lucas Vinicius Magalhães Pinheiro - 17/0061001 - lucasvini1298@gmail.com

Artur Filgueiras Scheiba Zorron - 18/0013696 - tucazorron@gmail.com

9 de novembro de 2020

## 1 Abstrato

Nesse trabalho iremos usar da técnica de classificação de dados de Florestas Randômicas para checar quais dados mais relevantes para o diagnóstico de COVID-19, baseado em dados anônimos de pacientes recebidos no Hospital Israelita Albert Einstein, São Paulo, Brasil.

O trabalho foi desenvolvido na linguagem *Python*.

## 2 O Problema

Com uma planilha contendo dados de exames feitos em pacientes recebidos no hospital mencionado e seus diagnósticos, precisamos criar um programa que consiga lançar um diagnóstico o mais preciso possível baseado nos *inputs* de um novo paciente.

## 3 Teoria

Para esse trabalho, o tipo de algoritmo usado é *Random Forests*, ou Florestas Randômicas em português. Para explicar sobre esse algoritmo precisamos primeiramente falar sobre Árvores de Decisão:

### 3.1 Árvores de Decisão

Uma Árvore de Decisão é um algoritmo de categorização de entradas, a partir de "perguntas" pertinentes às informações das entradas. Uma Árvore de Decisão procura padrões entre as características contidas em uma entrada e a sua relação com a saída ou "diagnóstico", e dá mais importância às características que melhor predizem a saída.

Cada divisão da árvore será uma pergunta que separe alguma das características das entradas em dois ou mais grupos, e em cada ramificação gerada, será recalculado qual das características restantes dentro daquele conjunto predizem com mais precisão a saída, até que todas as características tenham sido checadas.

Como exemplo, se temos um algoritmo que precisa catalogar entradas entre "cobra", "pepino" ou "cabo de vassoura", e em cada entrada temos as características "é verde", "comprimento", "se move" e "é um ser vivo".

Baseando nessas características, a informação "é verde" não consegue prever nada, pois tanto uma cobra, quanto um pepino quanto um cabo de vassouras podem ser verdes. Entretanto, a informação "se move" prediz bem se a entrada é ou não uma cobra, então a primeira pergunta da nossa árvore pode ser "o item se move?". Se respondermos sim, podemos garantir com certo grau de certeza que é uma cobra, e se não, podemos usar as outras informações para deduzir se é um pepino ou um cabo de vassoura.

Outra informação bastante relevante é "é um ser vivo", com essa informação podemos assumir com certo grau de certeza que o objeto é ou não um cabo de vassoura. Mas então qual das duas seria a melhor pergunta a ser aplicada na primeira divisão da Árvore de Decisões?

Assim como nesse exemplo, na vida real os casos não são tão óbvios de se descobrir qual informação é a mais pertinente, por isso é preciso fazer um treinamento da nossa Árvore de Decisões. Ou seja, precisamos dar à nossa Árvore de Decisões algumas entradas já com a saída para que ela possa checar quais combinações de informações podem gerar uma "cobra", um "pepino" ou um "cabo de vassoura", e para se calcular qual a informação mais pertinente, podemos usar o Índice Gini de pureza.

O Índice de Gini calcula, para cada informação das entradas, quão bem ela prevê a saída. O cálculo consiste em checar para cada possível valor da informação o quanto que ele consegue influenciar na saída final, e atribui para cada informação uma porcentagem. Após fazer esse cálculo para todas as informações, é escolhida a informação que gerou a menor porcentagem (menor grau de impureza) e essa informação será a pergunta da primeira divisória. Em seguida isso é repetido para cada ramificação, levando em conta somente as entradas que passaria pela ramificação específica.



Figura 1: Exemplo de uma Árvore de Decisão para checar se é ou não um bom momento para levar seu pet para passear.

Mas um problema existe: As árvores de decisão não conseguem ser 100% precisas, na grande maioria dos casos vai ter algum tipo de impureza na árvore gerada. Mas se construíssemos uma segunda árvore de decisões com diferentes ordens nas perguntas feitas, podemos diminuir essa

impureza. E se adicionarmos mais uma, e mais uma, e mais uma... Nós conseguimos uma Floresta Randômica. Floresta Randômica nada mais é do que um conjunto de várias Árvores de Decisão diferentes que catalogam as mesmas entradas.

## 4 Desenvolvimento

Para o desenvolvimento desse trabalho nós usamos a biblioteca de *python* "*sklearn*", uma biblioteca que cuida de aprendizado de máquina, e a biblioteca "*pandas*", que cuida do gerenciamento de arquivos dentro de um programa *python*.

Primeiramente precisamos importar o *dataset*, que está disponível como planilha *excel*, em uma matriz do tamanho da planilha do *excel*. Isso é para que possamos acessar cada coluna e cada fileira do *dataset* individualmente e para que possamos ter acesso direto a uma célula qualquer do *dataset*.

Com essa matriz feita, é necessário agora fazer o tratamento dos dados da matriz. Ao ser lido para *python*, as informações contidas no *dataset* são registradas como *string*, um tipo que não é suportado pela função do *sklearn* para montagem da floresta, uma vez que cálculos precisam ser feitos. Então precisamos converter todos os dados para *float*.

Outro problema com os dados é o fato de algumas das células estarem sem nenhum valor. Algumas soluções para isso são:

- 1 - Ignorá-las
- 2 - Atribuir a elas a média dos valores gerais da dita característica
- 3 - Atribuir a elas a moda dos valores gerais da dita característica

Com os dados normalizados, o próximo passo é alimentar o algoritmo com a matriz de dados relevantes, ou seja, tirando os dados de ID, diagnóstico e com as células vazias alteradas ou removidas, e uma *string* com os diagnósticos dos dados de entrada.

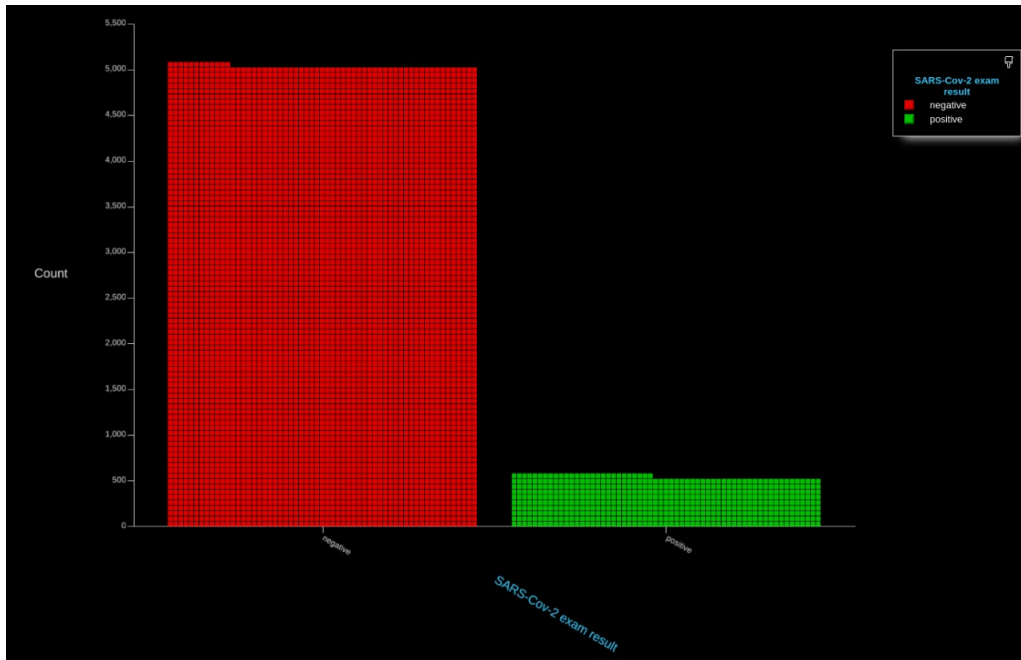


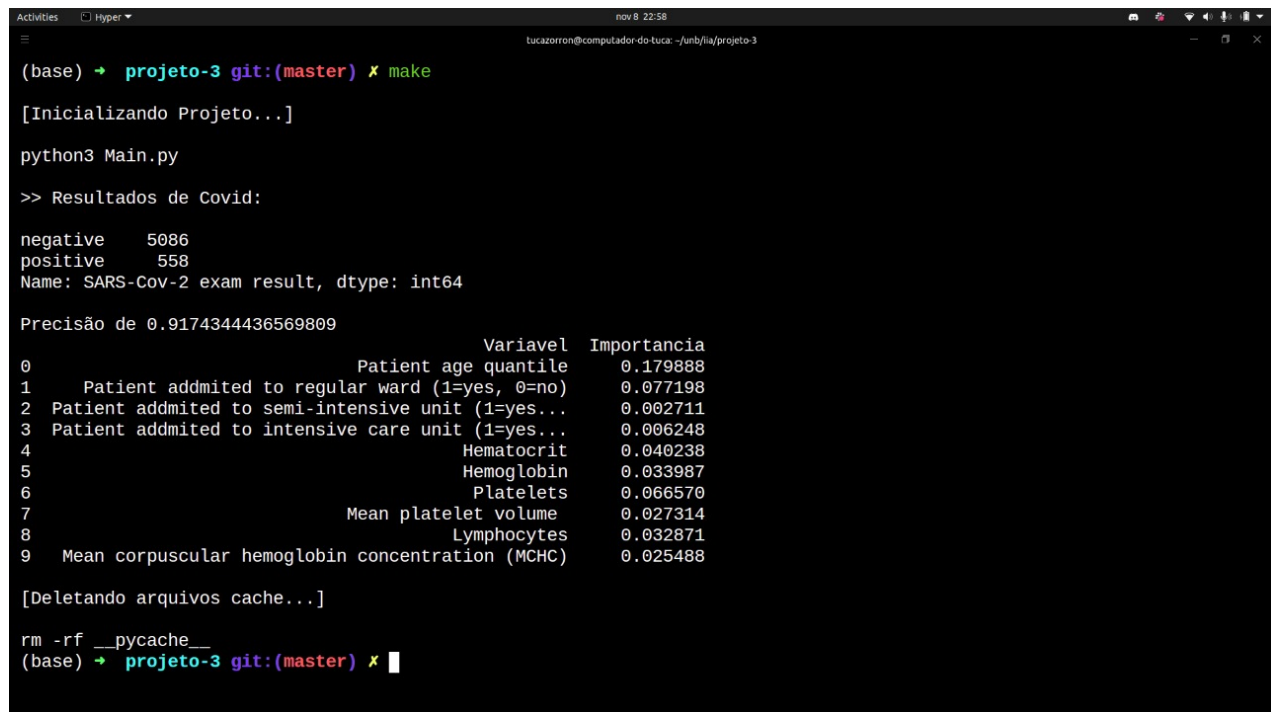
Figura 2: Diagnósticos visualizados em gráfico

## 5 Questionamentos

### 5.1 Pergunta 1

*”Baseando-se nos dados de laboratório (sem PCR) é possível predizer se o paciente estava positivo ou negativo para COVID-19? Com qual precisão? Quais os 10 (dez) testes/variáveis que são mais relevantes nessa resposta?”*

Para checar a precisão da nossa predição, nós usamos uma função do *sklearn* chamada *accuracy\_score*. Para essa função nós passamos como parâmetros uma *array* contendo os valores de saída e o modelo preditivo gerado pela nossa floresta randômica. Abaixo se encontra a saída gerada pelo algoritmo, mostrando a porcentagem de precisão e os 10 parâmetros mais relevantes.



```
(base) → projeto-3 git:(master) ✕ make
[Inicializando Projeto...]
python3 Main.py
>> Resultados de Covid:
negative      5086
positive       558
Name: SARS-Cov-2 exam result, dtype: int64

Precisão de 0.9174344436569809

```

	Variavel	Importancia
0	Patient age quantile	0.179888
1	Patient addmitted to regular ward (1=yes, 0=no)	0.077198
2	Patient addmitted to semi-intensive unit (1=yes...	0.002711
3	Patient addmitted to intensive care unit (1=yes...	0.006248
4	Hematocrit	0.040238
5	Hemoglobin	0.033987
6	Platelets	0.066570
7	Mean platelet volume	0.027314
8	Lymphocytes	0.032871
9	Mean corpuscular hemoglobin concentration (MCHC)	0.025488

```
[Deletando arquivos cache...]
rm -rf __pycache__
(base) → projeto-3 git:(master) ✕
```

Figura 3: Precisão e parâmetros mais relevantes

### 5.2 Pergunta 2

*”Baseando-se nos dados de laboratório é possível predizer quais casos, e por conta de quais variáveis/testes os pacientes teriam que ser: 1) internados em enfermaria, 2) internados em unidade semi-intensiva, 3) internados em unidade intensiva, 4) acompanhados em casa?”*

### 5.2.1 1) Tratamento Intensivo:

```
nov8 23:24
lucacaron@computador-do-luca: ~/lib/ia/projeto-3

>> Pacientes em unidades de tratamento intensivas:

0    5603
1      41
Name: Patient addmitted to intensive care unit (1=yes, 0=no), dtype: int64
4

Precisão de 0.9978738483345145

   Variavel  Importancia
0 Patient age quantile    0.033039
1 CovidBinario    0.014081
2 Patient addmitted to semi-intensive unit (1=yes... 0.003553
3 Patient addmitted to regular ward (1=yes, 0=no) 0.002781
4 Hematocrit    0.036321
5 Hemoglobin    0.056336
6 Platelets    0.087174
7 Mean platelet volume    0.025500
8 Lymphocytes    0.062753
9 Mean corpuscular hemoglobin concentration (MCHC) 0.037540
```

### 5.2.2 2) Tratamento Semi-Intensivo:

```
nov8 23:23
lucacaron@computador-do-luca: ~/lib/ia/projeto-3

>> Pacientes em unidades de tratamento semi-intensivas:

0    5594
1      50
Name: Patient addmitted to semi-intensive unit (1=yes, 0=no), dtype: int64
4

Precisão de 0.9985825655563431

   Variavel  Importancia
0 Patient age quantile    0.041422
1 CovidBinario    0.004355
2 Patient addmitted to regular ward (1=yes, 0=no) 0.005337
3 Patient addmitted to intensive care unit (1=yes... 0.007095
4 Hematocrit    0.053605
5 Hemoglobin    0.052601
6 Platelets    0.034997
7 Mean platelet volume    0.039860
8 Lymphocytes    0.041667
9 Mean corpuscular hemoglobin concentration (MCHC) 0.038755
```

### 5.2.3 1) Enfermaria:

```
nov8 23:23
lucacaron@computador-do-luca: ~/lib/ia/projeto-3

>> Pacientes em enfermaria normal:

0    5565
1      79
Name: Patient addmitted to regular ward (1=yes, 0=no), dtype: int64
4

Precisão de 0.9964564138908576

   Variavel  Importancia
0 Patient age quantile    0.047045
1 CovidBinario    0.045738
2 Patient addmitted to semi-intensive unit (1=yes... 0.006584
3 Patient addmitted to intensive care unit (1=yes... 0.002004
4 Hematocrit    0.050993
5 Hemoglobin    0.034944
6 Platelets    0.070474
7 Mean platelet volume    0.028713
8 Lymphocytes    0.052850
9 Mean corpuscular hemoglobin concentration (MCHC) 0.029752
```