

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA

INF01017 Aprendizado de Máquina

Trabalho 1: Aprendizado Ensemble com Árvores Aleatórias

Augusto Bennemann

Fabício Martins Mazzola

Porto Alegre, 10 de Maio de 2018

Introdução

Neste trabalho exploramos um aprendizado *ensemble*, por meio de florestas aleatórias. Para treinar o algoritmo e analisar seu desempenho utilizamos os quatro *datasets* propostos.

Inicialmente, fizemos um pré-processamento nos conjuntos de dados para padronizá-los de acordo com o formato de entrada que especificamos para nosso programa. Essas alterações foram, basicamente, incluir o nome das colunas na primeira linha e mover a coluna da classe para a última posição.

O algoritmo foi desenvolvido na linguagem Python 3. Junto a esse relatório há a versão final do código (*trabalho_1_adm.py*) e um arquivo do Jupyter (*Trabalho_1_ADM_iypnb.ipynb*) usado no desenvolvimento, que contém as mesmas funcionalidades do primeiro, mas em uma ordem um pouco diferente e com alguns testes individuais extras.

A validação cruzada estratificada foi realizada com $k = 10$ folds, conforme sugerido. Para o *bootstrap*, utilizamos 0.66 (66%) na seleção para o conjunto de treinamento. Para o número de atributos amostrados (m) utilizamos a raiz quadrada do número total de atributos, conforme valor padrão sugerido pela literatura.

Estruturas de dados

Para implementar a Árvore de Decisão, criamos a classe *Node*, que define um nodo que pode ter outros nodos descendentes. No caso, há um nó raiz do qual derivam, em algum nível, todos os outros nós da árvore.

Cada nó possui: um atributo (seu nome, a saber “*Temperatura*”), uma *classType* (a classe prevista quando é um nó folha, ou *None* caso contrário), um operador da linguagem (*==*, *<=* ou *>=*); e descendentes.

```
class Node(object):
    def __init__(self, attribute=None):
        self.attribute = attribute
        self.classType = None
        self.operator = operator.eq
        self.descendents = {}

    def addBranch(self, subAttribute, op, node):
        self.descendents[subAttribute] = node
        self.descendents[subAttribute].updateOperator(op)

    def updateClass(self, classType):
        self.classType = classType

    def updateAttribute(self, attribute):
        self.attribute = attribute

    def updateOperator(self, op):
        self.operator = op
```

Definição da classe *Node*

```

DiabetesPedigreeFunction: None
-> 0.16819
  BMI: None
  -> 0.46661
    Age: None
    -> 0.18263
      Glucose: None
      -> 0.53588
        Insulin: None
        -> 0.03798
          SkinThickness: None
          -> 0.09804
            None: 0
            -> 0.09803
              Pregnancies: None
              -> 0.12046
                None: 0
                -> 0.12045
                  BloodPressure: None
                  -> 0.52323
                    None: 0
                    -> 0.52322
                      None: 0
                      -> 0.03797
                        None: 0
                        -> 0.53587
                          None: 0
                          -> 0.18262
                            None: 0
                            -> 0.4666
                              None: 0
                              -> 0.16818
                                None: 0

```

Exemplo de árvore induzida para o conjunto de dados Diabetes

Corretude

A fim de provar a corretude, executamos o algoritmo de indução da árvore no conjunto de dados de *benchmark* fornecido. À direita, estão os valores obtidos, que conferem com o esperado.

```

Ganho para Tempo: 0.247
Ganho para Temperatura: 0.029
Ganho para Umidade: 0.152
Ganho para Ventoso: 0.048

```

Inicialmente, implementamos uma versão básica do algoritmo de indução, usando apenas o Ganho de Informação como critério de seleção. A seguir, à esquerda, está a árvore gerada nesse caso e os valores de Ganho de Informação a cada divisão de nós. Em

seguida, inserimos no algoritmo a seleção aleatória de m atributos. À direita está uma indução realizada por essa versão - que é a final -, também com os valores de Ganho a cada divisão.

```

Tempo: None
-> Ensolarado
  Umidade: None
  -> Alta
    None: Nao
  -> Normal
    None: Sim
-> Nublado
  None: Sim
-> Chuvoso
  Ventoso: None
  -> Falso
    None: Sim
  -> Verdadeiro
    None: Nao

```

```

Ganho para Temperatura: 0.029
Ganho para Ventoso: 0.048
Ganho para Umidade: 0.152
Ganho para Tempo: 0.247

Ganho para Ventoso: 0.02
Ganho para Temperatura: 0.571
Ganho para Umidade: 0.971

Ganho para Temperatura: 0.02
Ganho para Ventoso: 0.971

```

```

Umidade: None
-> Alta
  Ventoso: None
  -> Falso
    Tempo: None
    -> Ensolarado
      None: Nao
    -> Nublado
      None: Sim
    -> Chuvoso
      None: Sim
  -> Verdadeiro
    Temperatura: None
    -> Quente
      None: Nao
    -> Amena
      None: Sim
-> Normal
  None: Sim

```

```

Ganho para Ventoso: 0.048
Ganho para Umidade: 0.152

Ganho para Ventoso: 0.02
Ganho para Temperatura: 0.02

Ganho para Temperatura: 0.0
Ganho para Tempo: 1.0

Ganho para Temperatura: 0.252

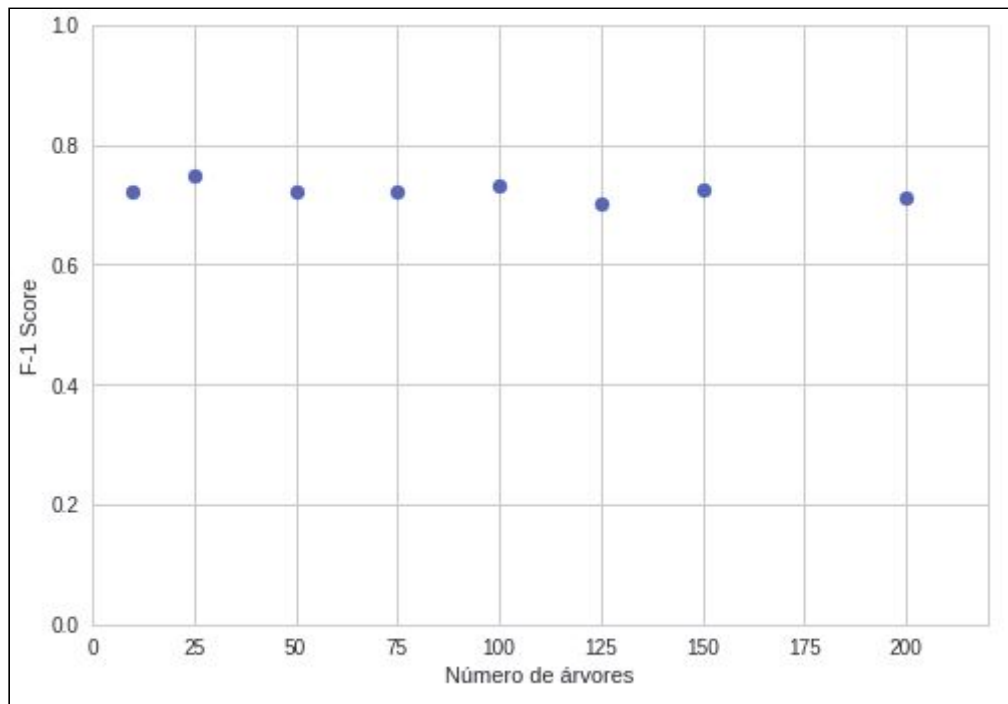
```

Análise de desempenho

Para avaliar o desempenho do algoritmo, testamos os quatro conjunto de dados variando o parâmetro *ntree* (número de árvores) para 10, 25, 50, 75, 100 e 125. Como métrica, para os três primeiros foi utilizada a F1-measure, enquanto que para o último utilizou-se a “média da precisão macro” pois ele possui 3 classes.

1. Diabetes

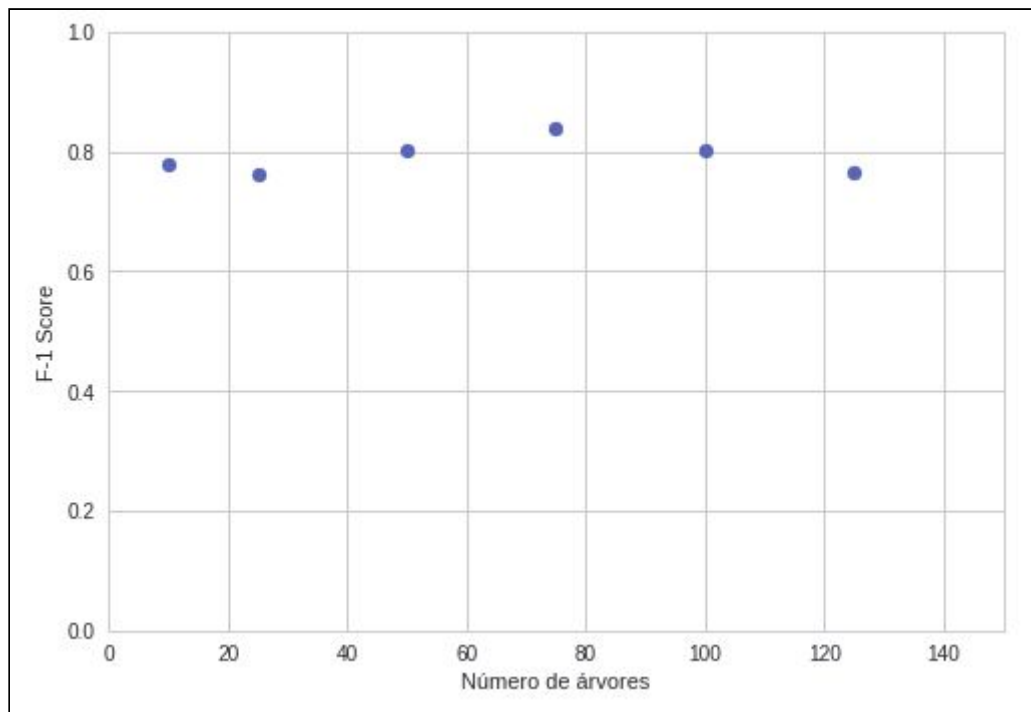
F1 \ Árvores	10	25	50	75	100	125	150	200
Média	0.72	0.74	0.72	0.72	0.73	0.70	0.73	0.72
Desvio	0.05	0.03	0.05	0.04	0.04	0.03	0.03	0.04



O desempenho do algoritmo para esse conjunto de dados variou muito pouco. Ademais, ele variou em ambas direções, sem seguir uma progressão, portanto não está evidente se a variância de ntree teve alguma relevância estatisticamente ou se as diferenças são apenas efeito da aleatoriedade.

2. Ionosphere

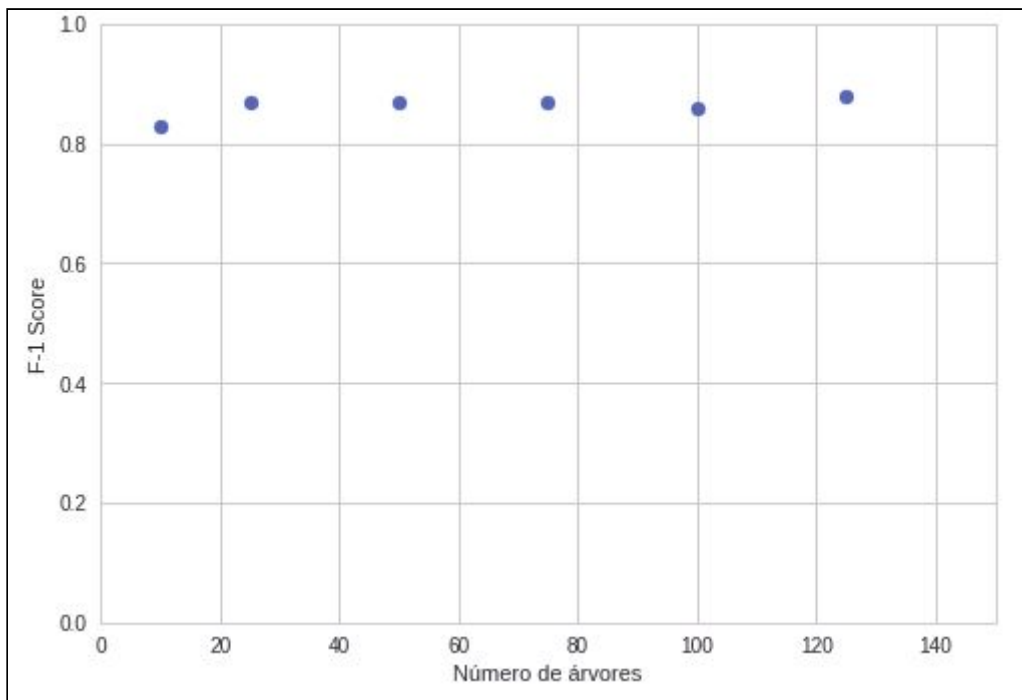
F1 \ Árvores	10	25	50	75	100	125
Média	0.78	0.76	0.80	0.84	0.80	0.76
Desvio	0.05	0.04	0.06	0.07	0.06	0.04



Esse conjunto de dados foi o que apresentou a maior variação. Ele progride de forma relativamente acentuada à medida que *n_{tree}* cresce até 75. No entanto, depois decai.

3. Breast Cancer (Extra)

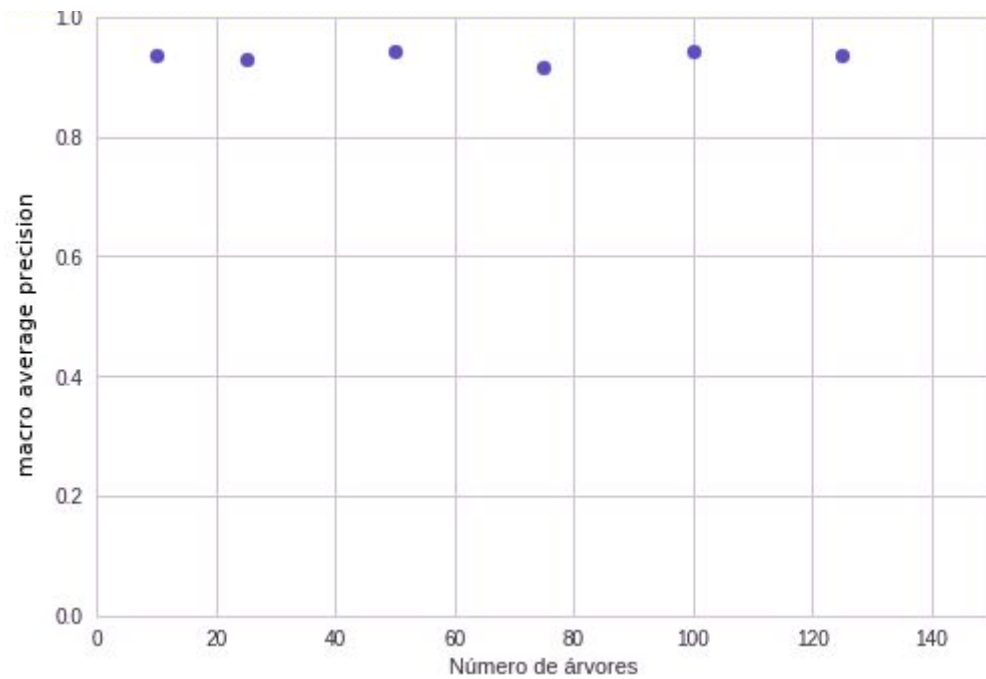
F1 \ Árvores	10	25	50	75	100	125
Média	0.83	0.87	0.87	0.87	0.86	0.88
Desvio	0.07	0.03	0.04	0.05	0.05	0.05



Os resultados para esse conjunto de dados mostram uma progressão contínua, apesar de pequena.

4. Wine

prec_macro \ Árvores	10	25	50	75	100	125
Média	0.94	0.93	0.94	0.92	0.94	0.94
Desvio	0.06	0.07	0.04	0.06	0.04	0.07



Esse conjunto de dados foi o que apresentou a melhor métrica. No entanto, varia muito pouco com as variações de *ntree*.

Conclusão

Teoricamente, à medida que *ntree* cresce o erro deve diminuir e, consequentemente, a média de acertos aumentar. No entanto, no geral, isso não ficou muito nítido nos resultados obtidos.

Ademais, as variações nas métricas foram muito pequenas, a ponto de não se entender o quanto elas são efeito da variância do parâmetro ou se ocorrem na maior parcela devido à aleatoriedade do algoritmo. Também, por conta disso, não ficou evidente se essas diferenças são estatisticamente relevantes.

Esperávamos encontrar uma diferença maior nos resultados. Independente disso, realizar esse trabalho foi bastante interessante e também importante, pois contribuiu para compreendermos mais a fundo vários conteúdos visto em sala de aula.