Aprendizado Ensemble - Florestas Aleatórias
Trabalho 1 – INF01017/CMP263 – 2018/2

Cristian Leal Nornberg[1] , Filipe Lins[1], Pablo Pavan[1]

[1] Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

clnornberg@inf.ufrgs.br, fmlins@inf.ufrgs.br, pablo.pavan@inf.ufrgs.br

Resumo. Os métodos de classificação baseados em árvores de decisão representam modelos simples, porém muito utilizados na aprendizagem de máquina. Isso deve-se principalmente a expressividade do modelo, por ser intuitivo e de fácil aplicação. Este trabalho tem o objetivo de implementar o algoritmo de Florestas Aleatórias para tarefas de classificação, implementar o algoritmo de indução da árvore utilizando o ganho de informação para seleção dos atributos, treinar a árvore de decisão e realizar a classificação de uma nova instância. Foi utilizado o mecanismo de bootstrap para gerar subconjuntos a partir dos dados de treinamento, e o desempenho do modelo foi verificado utilizando a técnica de validação cruzada estratificada. Além disso, foi avaliado o impacto do número de árvores no desempenho do ensemble.

1. Introdução

Os métodos de classificação baseados em árvores de decisão representam modelos simples, porém muito utilizados na aprendizagem de máquina. Isso deve-se principalmente a expressividade do modelo, por ser intuitivo e de fácil aplicação. As árvores de decisão são modelos hierárquicos capazes de guiar a tomada de decisão sobre qual classe pertence uma determinada instância. As árvores representam um conjunto de regras consistentes de classificação SE-ENTÃO. Para um exemplo de entrada, são realizados testes dos valores de cada atributo contra a árvore de decisão, e a sequência de testes e suas respectivas respostas formam um caminho na árvore, partindo da raiz até uma determinada folha (nó terminal).

A classe associada a folha é retornada como a classe predita para o exemplo de entrada. É utilizada a estratégia de divisão e conquista, onde o problema original é dividido em subproblemas mais simples, aplicando recursivamente esta estratégia. Escolhe-se um atributo para adicionar à árvore, e estende-se a árvore adicionando um ramo para cada valor do atributo selecionado para dados categóricos ou determina-se um ponto de corte para dados contínuos. Os dados são então divididos em partições conforme valores do atributo testado, um para cada ramo selecionado. Este procedimento é repetido recursivamente para cada partição de exemplos até atingir um critério de parada, que pode ser quando todos exemplos na partição resultante pertencem a mesma classe, não há mais atributos para escolher ou uma partição resultante é vazia. Então cria-se uma folha com a classe correspondente ou de acordo com a classe majoritária.

2. Implementação

Para a implementação da árvore de decisão baseada em ganho de informação (algoritmo ID3), foi utilizado a linguagem de programação *Python3*, juntamente com algumas bibliotecas para a manipulação dos dados. A linguagem Python foi utilizada, pela facilidade na manipulação dos dados e a fácil implementação de estruturas de dados. As principais bibliotecas utilizadas foram:

- Pandas[1]: Biblioteca open-source que oferece suporte a estruturas e ferramentas de análise de dados para Python, como manipulação, leitura e visualização de dados.
- Numpy[2]: Biblioteca de computação científica que oferece suporte a manipulação de dados e arrays com múltiplas dimensões.
- Math: Biblioteca que fornece acesso à funções matemáticas.

Para a implementação da estrutura da árvore, utilizamos uma classe para representar os nós, sendo que a árvore é uma lista de objetos desta classe. A classe possui 4 métodos, o método de inicialização (__init__) da classe, onde toda a criação dos nós ocorrem de forma recursiva, o método de representação (__repr__) e o método de (__str__) que são utilizados para o print da árvore, e o último método é o de classificação de novas instâncias (evaluate).

Os atributos escolhidos para representar os nós foram foram 8, sendo que temos mais dois atributos para a manipulação dos dados. Os atributos de representação dos nós são:

- inference_att: Nome da aresta que criou este nó (atributo selecionado).
- result_label: Se este atributo não for *None* este nó representa uma folha, e este atributo contém o resultado.
- parent: Este atributo faz referência ao nome do nó pai do nó atual.
- gain: Armazena o ganho de informação que gerou a criação deste nó
- level: Armazena o nível deste nó na árvore.
- name_att: Nome do nó atual (atributo selecionado).

Seguindo no desenvolvimento, escrevemos 22 funções para auxiliar na criação, classificação, e validação das árvores para serem testadas. Dentre estas funções, podemos destacar as principais, são elas:

- most_gain(dataset): Recebe como parâmetro o dataframe dos dados a serem analisados, retornando o index e o ganho do atributo selecionado.
- k_folds(dataset, i, k): Recebe como parâmetro o dataframe dos dados, o kfold que será usado para o teste e o número de folds que o dado deve ser dividido. Retorna o dataframe dos dados de treino com os kfodls-1 e o dataframe de teste com o fold selecionado.
- create_n_bootstraps_train(um_bootstraps, num_instances): Retorna uma lista com os valores das linhas que vão ser utilizadas na criação dos dados do bootstrap, recebe como parâmetro o número de bootstraps a serem criados e o número de instâncias do dataframe utilizado.

[1] https://pandas.pydata.org/
[2] http://www.numpy.org/

- bootstrap_n_data_list_train(numbootstraps, numinstances, dataset, list_of_bootstraps_train): Esta função, depende da função descrita acima pois recebe como parametro a lista retornada desta função (list_of_bootstraps_train). Os outros parametros são, número de bootstraps, número de instancias e o dataframe dos dados. Retorna um dataframe.
- f_measure(prec, rev, beta): Esta função recebe como parametros o retorno da função de precisão, da função de recall e o peso beta, retornando a F1-measure da random floresta testada.

Os teste foram executados em computador pessoal, composto por um processador i7-7700hq de 2,80 GHz de frequência com 16GB de memória RAM DDR4.

Para a geração dos gráficos foi utilizado a linguagem de programação R, juntamente com a biblioteca ggplot2. O gráfico escolhido para representar os resultados, foi o boxplot. O boxplot identifica onde estão localizados 50% dos valores mais prováveis, a mediana e os valores extremos. Junto com o boxplot foi plotado um ponto que representa a média dos valores. Cada teste foi realizado no mínimo 3 vezes.

3. Seleção de Atributos para Indução da Árvore de Decisão

Durante o processo de indução das árvores de decisão, o conjunto de dados foi dividido em conjunto de treinamento e testes utilizando a estratégia de **validação cruzada**, que consiste em uma técnica para avaliar a capacidade de generalização de um modelo, a partir de um conjunto de dados. Para esta tarefa, precisamos dividir os dados em um conjunto de treinamento e um conjunto de validação (teste), para isso utilizamos a técnica de k-fold que consistem em em dividir os dados em partes, k partes -1 são utilizadas para o treinamento e a parte restante para o teste. Isso é repetido até que todas as k partes foram utilizadas com teste. A Figura 1 representa um exemplo deste processo, onde o k é foi igual a 5, cada linha é uma iteração, e os blocos verdes são utilizados para treino e o bloco vermelho para teste.

Para cada dataset de treino, foi implementado um ensemble, para isso utilizamos a técnica de bootstrap com o objetivo de criar datasets do conjunto original de dados usando amostragem aleatória de instâncias com reposição, onde cada bootstrap foi utilizado para o treinamento de uma árvore. Neste trabalho, variou-se o número de árvores no ensemble, buscando verificar a performance, foi utilizado bootstraps de 1, 5, 10 e 20. A medida utilizada para avaliar a performance do foi a F1-measure, atribuindo o mesmo peso para precisão e revocação (β=1).

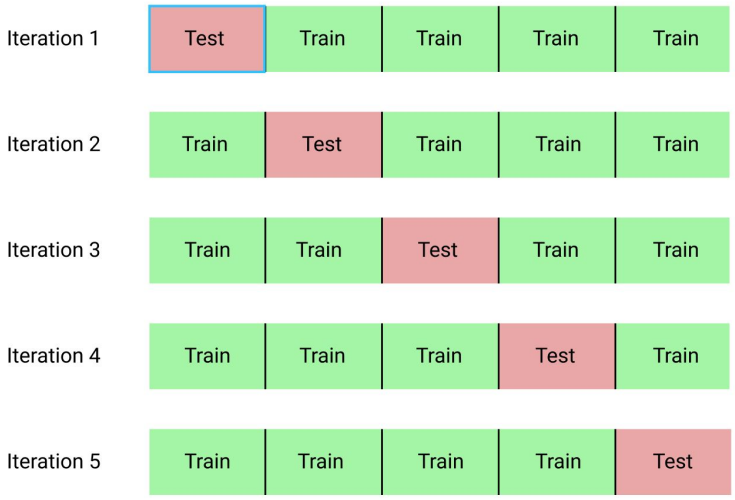

Figura 1 : Exemplo de funcionamento da técnica de validação cruzada utilizando a divisão de dados com k-fold - Fonte: https://my.oschina.net/Bettyty/blog/751627 (2018)

Com o objetivo de discretizar os atributos de valor numérico (valores contínuos), foi definido um ponto de corte, calculando-se a média aritmética dos valores da instância do conjunto de treinamento. O critério de seleção de atributos utilizado para a divisão dos nós foi o ganho de informação, baseado no conceito de entropia (grau de aleatoriedade) dos conjuntos.

4. Conjuntos de Dados e Resultados

No desenvolvimento do trabalho foram utilizados cinco conjunto de dados. O primeiro, é um conjunto básico para a inferência de uma árvore de decisão, utilizando ganho de informação, este conjunto serve somente para validar o algoritmo utilizado, já que o mesmo é um benchmark do estado da arte, muito utilizado para esta finalidade. Outros quatros são executados utilizando o aprendizado ensemble e a validação cruzada para demonstrar o desempenho.

4.1. Conjunto de dados de Benchmark

Formado por 4 atributos categóricos, 14 instâncias e duas classes. Joga é a classe a ser predita. Este conjunto de dados foi utilizado para verificar a implementação do algoritmo de indução da árvore de decisão. A Figura 2 apresenta a árvore para o conjunto de dados do Benchmark, cada linha representa um nó, sendo que em cada nó, representamos o nome da aresta que liga com ele com o seu nó pai, e o seu nome, no formato [aresta,[nome]]. Também são mostrados os níveis de cada nó e o ganho de informação que foi utilizado para a criação do mesmo. Caso o nó tenha o atributo resultado, este nó representa uma folha e o valor do resultado indica o que foi inferido.

```
ROOT [Tempo] -> level: 1 gain: 0.2467498197744391

[           CHILD [Falso,[Ventoso]] -> results: ['Sim'] level: 3 gain: 0.9709505944546686

,           CHILD [Verdadeiro,[Ventoso]] -> results: ['Nao'] level: 3 gain: 0.9709505944546686

,       CHILD [Chuvoso,[Ventoso]] -> level: 2 gain: 0.2467498197744391

,           CHILD [Alta,[Umidade]] -> results: ['Nao'] level: 3 gain: 0.9709505944546686

,           CHILD [Normal,[Umidade]] -> results: ['Sim'] level: 3 gain: 0.9709505944546686

,       CHILD [Ensolarado,[Umidade]] -> level: 2 gain: 0.2467498197744391

,       CHILD [Nublado,[Tempo]] -> results: ['Sim'] level: 2 gain: 0.2467498197744391

]
```

Figura 2: Árvore gerada para o conjunto de dados de benchmark.

4.2. Conjunto de dados Breast Cancer Wisconsin

Formado por 10 atributos numéricos, 699 instâncias e duas classes (4=malignant, 2=benign). O objetivo é predizer se um determinado exame médico indica ou não a presença de câncer. A Figura 3 apresenta os resultados da métrica F1-measure, na eixo y (valores estão em escala diferente) temos os valores desta métrica, e no eixo x temos o número da iteração do K-Fold utilizados.

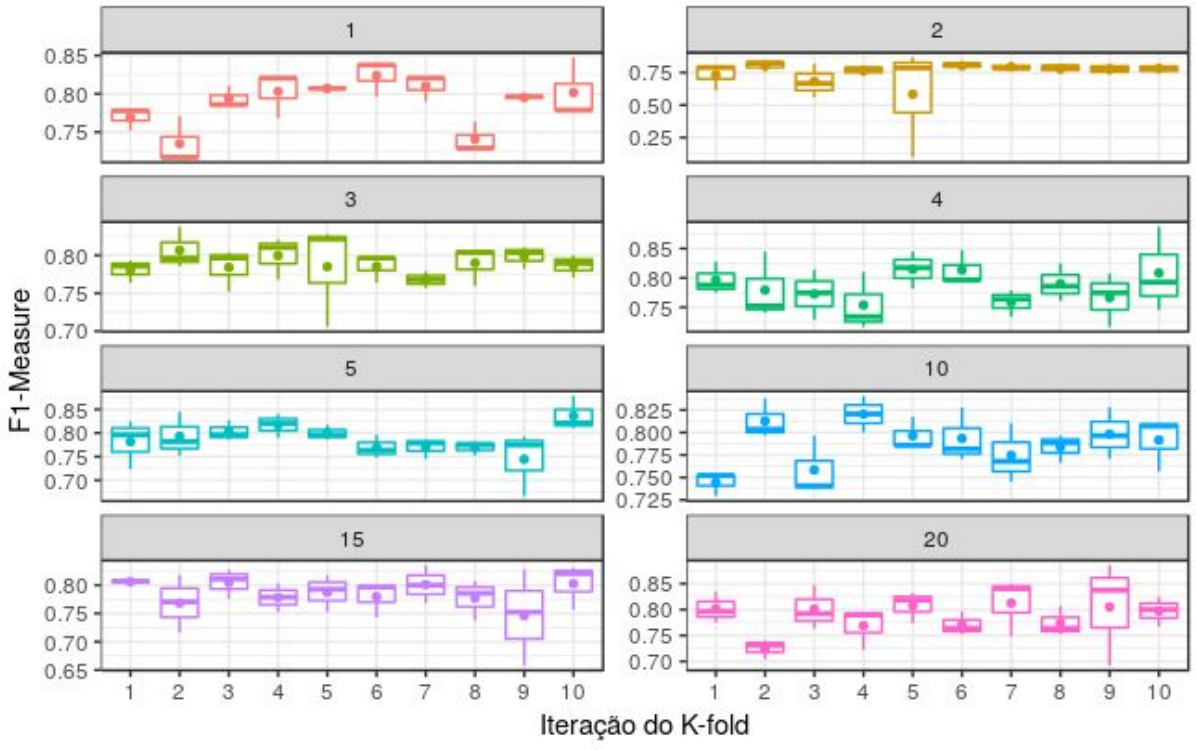

Figura 3: F1-Measure do dataset Breast Cancer Wisconsin com a variação do número de árvores utilizadas e a iteração do K-fold - Fonte: Autor (2018)

Cada faceta e cor representa o número de bootstraps utilizados que multiplicado pelo número de iterações, neste caso 10, temos o número de árvores utilizadas. O melhor caso, onde a F1-measure chegou a 0.8871 na iteração igual a 10 usando 4 bootstraps. O pior caso foi encontrado utilizando 2 bootstraps na iteração 5 onde a F1 ficou na casa de 0.09756. Em média o melhor caso foram com 3 e 5 bootstrap onde a F1 chegou a 0.7885.

Podemos notar que com o aumento do número de árvores, o desempenho não mostrou um aumento significativo (p-value < 0,05), diferente do bootstrap, em algumas iterações do K-fold, temos uma variação. Esta variação pode ser gerada na divisão dos dados, onde os dados gerados não representam bem as instâncias, e o uso de vários bootstrap não melhoram esta variação.

4.3. Conjunto de dados Wine

Formado por 13 atributos numéricos, 178 instâncias e 3 classes. O objetivo é predizer o tipo de um vinho baseado em sua composição química. A Figura 4 apresenta os resultados da métrica F1-measure, na eixo y temos os valores desta métrica, e no eixo x temos o número da iteração do K-Fold utilizados.

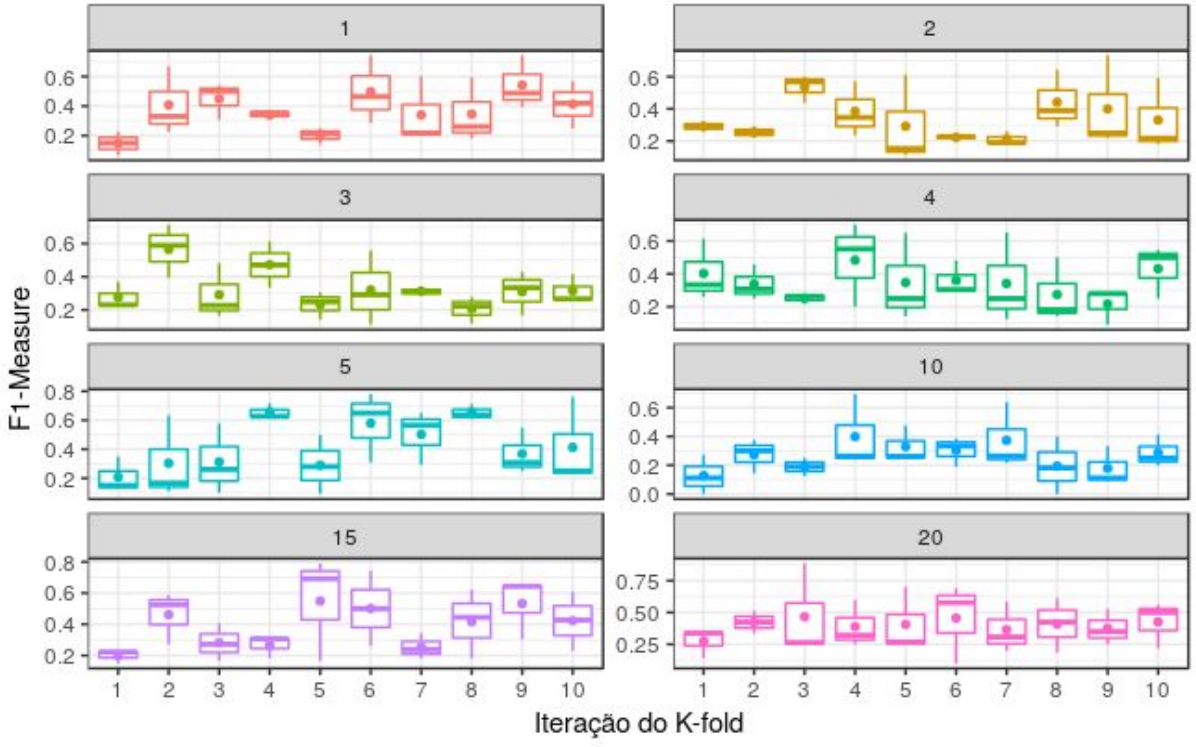

Figura 4: F1-Measure do dataset Wine com a variação do número de árvores utilizadas e a iteração do K-fold - Fonte: Autor (2018)

Como no gráfico anterior a faceta e a cor representa o número de bootstraps utilizados que multiplicado pelo número de iterações, neste caso 10, temos o número de árvores utilizadas. O melhor caso, onde a F1-measure chegou a 0.8852 ocorreu na iteração

3 usando 20 bootstrap. O pior caso foi encontrado utilizando 10 bootstrap na primeira iteração onde a F1 ficou na casa de 0.0000. Em média o melhor caso ocorreu com 5 bootstrap onde a F1 chegou a 0.42825.

4.4. Conjunto de dados Ionosphere

Formado por 34 atributos numéricos, 351 instâncias e 2 classes (g=good, b=bad). O objetivo é predizer se a captura de sinais de um radar da ionosfera é adequada para análises posteriores ou não. A Figura 5 apresenta os resultados da métrica F1-measure, na eixo y temos os valores desta métrica, e no eixo x temos o número da iteração do K-Fold utilizados.

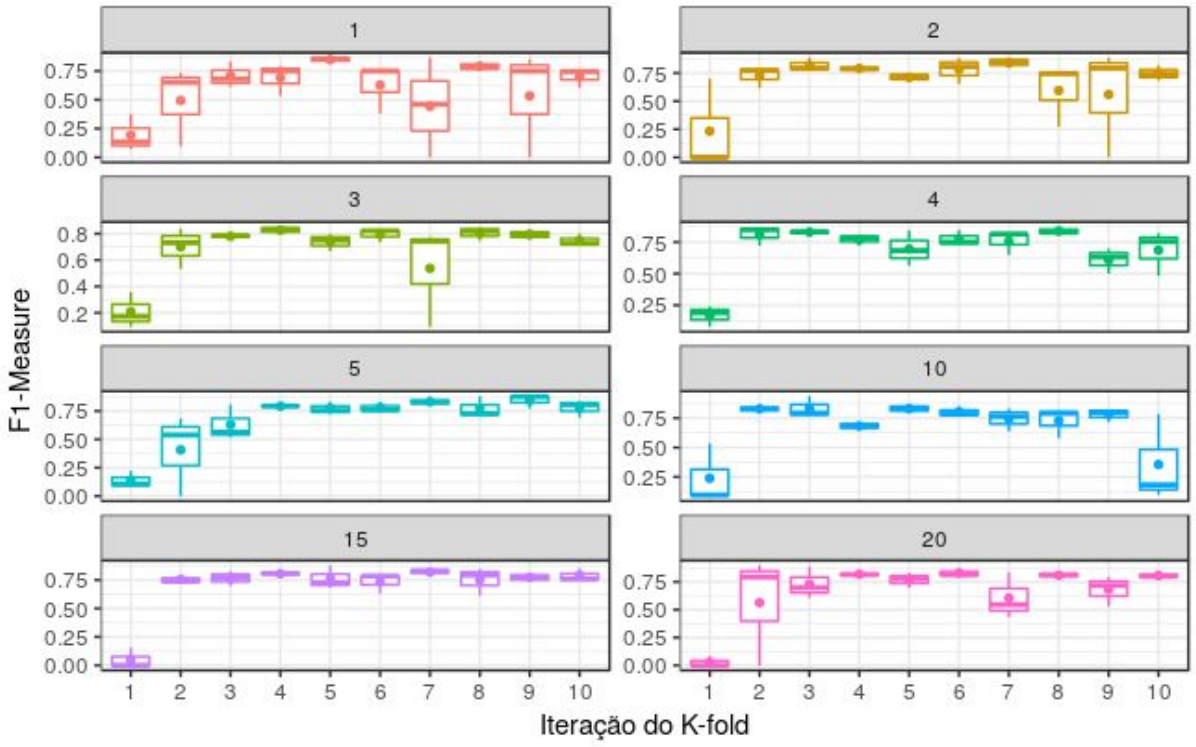

Figura 5: F1-Measure do dataset Ionosphere com a variação do número de árvores utilizadas e a iteração do K-fold - Fonte: Autor (2018)

Cada faceta e cor representa o número de bootstraps utilizados que multiplicado pelo número de iterações, neste caso 10, temos o número de árvores utilizadas. O melhor caso, onde a F1-measure chegou a 0.93750 na iteração 3 usando 10 bootstrap. O pior caso foi encontrado utilizando 1, 2, 5, 15 e 20 bootstrap sempre na iteração 1 onde a F1 ficou zerada, podendo indicar uma problema na divisão dos K-Folds para este dataset. Em média o melhor caso ocorreu com 15 bootstrap onde a F1 chegou a 0.7004.

4.5. Conjunto de dados Chess End-Game[3]

Esta base de dados possui um total de 3196 instâncias cujo formato é uma sequência de 37 atributos categóricos no qual cada instância descreve o cenário do final do jogo de xadrez. Os 36 primeiros atributos descrevem o tabuleiro e o último classifica se o jogo foi perdido ou ganho pelas peças brancas. Este dataset foi escolhido porque nos outros datasets os atributos são totalmente numéricos e neste caso os atributos são totalmente categórico. Assim este resultado demonstra como o modelo ensemble se comportou utilizando o ganho de informação para este tipo de atributo categórico.

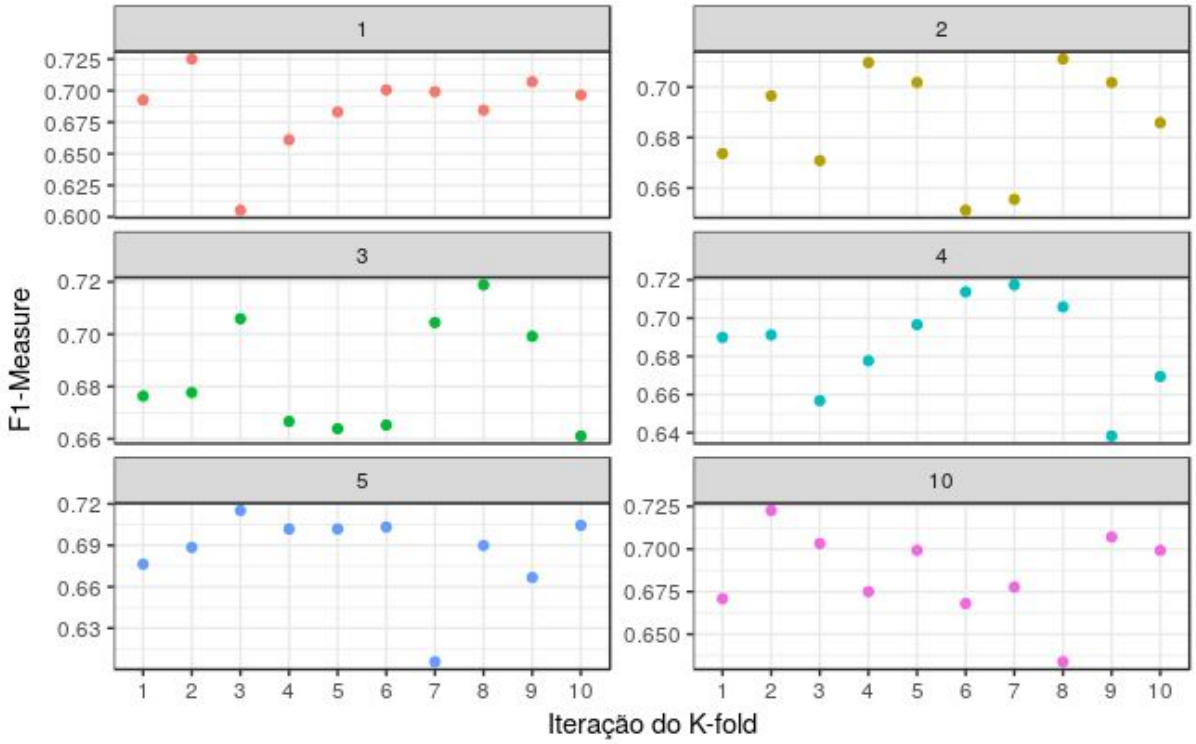

Figura 6: F1-Measure do dataset Chess End-Game com a variação do número de árvores utilizadas e a iteração do K-fold - Fonte: Autor (2018)

A Figura 6 apresenta os resultados da métrica F1-measure, na eixo y temos os valores desta métrica, e no eixo x temos o número da iteração do K-Fold utilizados. Cada faceta e cor representa o número de bootstraps utilizados que multiplicado pelo número de iterações, neste caso 10, temos o número de árvores utilizadas. O melhor caso, onde a F1-measure chegou a 0.7251 usando 1 bootstrap na segunda iteração do K-Fold este se manteve o melhor para diferentes quantidades de árvores. O pior caso foi encontrado utilizando 1 bootstrap na terceira iteração do K-Fold na qual a F1 chegou a 0.6053 percebeu-se que com o aumento de árvores houve um aumento da F1 para esta iteração do K-Fold. Em média o melhor caso ocorreu com o bootstrap 2 onde a F1 chegou a 0.6858.

4.6 Desempenho

[3] https://archive.ics.uci.edu/ml/machine-learning-databases/chess/king-rook-vs-king-pawn/

Outra análise realizada foi em relação ao tempo de execução, testamos para os 4 datasets. A Figura 7 apresenta um gráfico de relação do número de bootstrap utilizado (eixo x) e o tempo de execução em segundos (eixo y). Cada cor representa um dos 4 datasets. Foi utilizado 1, 5, 10, 20, 30 e 50 bootstrap exceto para o dataset Chess End-Game onde foi utilizado somente 1,5 e 10 pelo fato do tempo de execução ser muito maior que dos outros dataset.

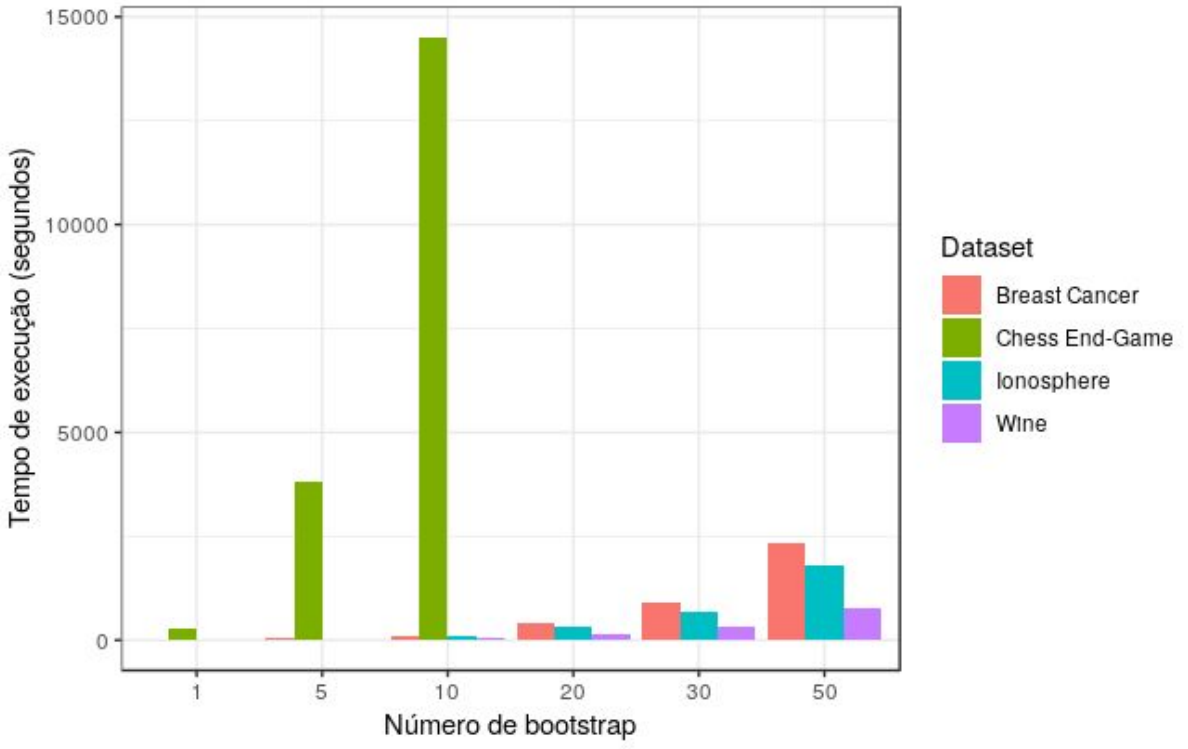

Figura 9: Tempo de execução em segundos variando o número de árvores utilizadas

Podemos notar que o Chess End-Game é o dataset que consome mais tempo de execução, com 10 bootstrap comparado com os outros ele é 181 vezes mais lento, chegando há 14.519 segundos. Outro ponto analisado é que no caso do outros 3 datasets: Breast Cancer; Ionosphere e Wine, nestes casos o tempo de execução é proporcional a F1-measure, onde quanto mais tempo de execução, melhor é a métrica.

6. Execução do Algoritmo

A fim de executar os diferentes datasets é necessário executar os seguintes arquivos python breasc-cancer_scratchmain.py, wine_scratch.py, ionosphere_scratch.py, dadosBenchmark_scratchmain.py, chess_scratch.py que recebe como argumento o número de bootstrap para rodar o algoritmo estes programas têm como saída as métricas de cada iteração do K-Fold e no final a média, a variância e o desvio padrão de todos os resultados.

O seguinte programa scratch_dadosbasictest.py. imprime a árvore do dadosBenchmark_validacaoAlgoritmoAD.csv, porém é necessário trocar o valor do enable no arquivo gain.py na função most_gain na linha 82 para False para obter a árvore sem considerar o item do m-atributos que gera uma árvore diferente do esperado do benchmark.

7. Conclusão

O trabalho buscou apresentar os principais resultados encontrados no desenvolvimento de um ensemble, utilizando técnicas como bootstrap e validação cruzada. Os resultados apresentados exploram um benchmark para validação do algoritmo e 4 datasets para realizar a métrica de desempenho, neste trabalho optou-se pela F1-measure.

Os principais resultados alcançados mostram que não obtivemos melhor desempenho quando utilizado maiores números de árvores, sendo que o máximo de desempenho encontrado foi utilizando o dataset Breast Cancer, onde a F1-measure chegou na casa de 0.7885 em média. Outro ponto encontrado foi que o mecanismo de amostragem de m atributos para a cada divisão de nó, a partir dos quais será selecionado o melhor atributo de acordo com o critério de Ganho de Informação acaba aumentando o números de nós da árvore e diminuindo o números de níveis por consequência conseguindo generalizar melhor a classificação dos dados.