**INSTITUTO TECNOLÓGICO DE AERONÁUTICA - ITA**

DIVISÃO DE CIÊNCIA DA COMPUTAÇÃO

**CTC-17 – Inteligência Artificial**


# Projeto III – Aprendizado de Máquina

- **Classificador de filmes**

**ALUNO**

Felipe Tuyama de Faria Barbosa ftuyama@gmail.com

**PROFESSOR**

Paulo André Lima de Castro pauloac@ita.br

São José dos Campos, 28 de Outubro de 2016

# 1. INTRODUÇÃO

O objetivo deste projeto é implementar algoritmos de inteligência artificial envolvendo aprendizado supervisionado de máquina. A partir de uma base de dados de um milhão de classificações de filmes (uma a cinco estrelas) conferidas por usuários diversos do MovieLens, será implementado um classificador baseado em árvore de decisão. Em um segundo momento o resultado será comparado com um classificador a priori.

As soluções foram implementadas na linguagem de alto nível Python 2.7, devido à facilidade de prototipação e desenvolvimento.

# 2. DESCRIÇÃO

O aprendizado de máquina utilizando árvore de decisão consiste primariamente em, a partir de uma base de dados de conhecimento previamente concedida, construir uma árvore de decisão com as informações daquela base de conhecimentos. Assim, as avaliações de filmes realizadas por usuários possuem muitas informações sobre quem avalia o filme, sendo possível reconstituir padrões em uma estrutura de dados (árvore de decisões).

Cada nó da árvore representa uma decisão de atributo (teste de propriedade), os ramos são os valores de teste (decisões) e a cada folha é o resultado daquele classificador para uma dada pessoa (ou conjunto de atributos). São assim 4 etapas para o classificador: leitura da base de dados, organização ou construção da árvore de decisões, treinamento do classificador e enfim a navegação na árvore de decisões, gerando ao final uma classificação do filme ao usuário.

Neste trabalho será descrita a implementação da árvore de decisão usando o algoritmo ID3, posteriormente treinando-a com um conjunto de dados a partir da validação cruzada e finalmente testada a sua performance em relação a uma classificação a priori direta a partir da base de dados (cálculo da média de classificação).

# 3. IMPLEMENTAÇÃO

## 3.1 Classificador a priori

Inicialmente, foi realizada a leitura sequencial dos arquivos de classificação de filmes, usuários e filmes da base de dados, convertendo-os para uma estrutura em forma de mapa, facilitando o acesso aos dados:

```
# Ratings:  [{movie: [(user, rate, movie)]}]
# Users:    [{user:  [gender, age, occupation, zipcode]}]
# Movies:   [{movie: [title, genre]}]
```

**Figura 1 – Estrutura de dados para armazenar informações lidas.**

Para implementar o classificador a priori, basicamente implementei uma função para mapear o conteúdo das linhas nesta estrutura e então reduzir a informação para um único mapa de classificação (contabilizar a classificação de cada filme, calculando a média). O algoritmo usado foi MapReduce:

```
def map(lines):
    u"""Contabiliza avaliações."""
    map = {}
    # Map
    for line in lines:
        (_, movie, rank, _) = line.split("::")
        if not (movie in map):
            map[movie] = [0, 0, 0, 0, 0, 0]
        map[movie][int(rank)] += 1
    # Reduce
    for movie in map:
        total = sum(map[movie])
        for i in range(1, 6):
            map[movie][0] += i * map[movie][i]
        map[movie][0] = map[movie][0] / total
    return map
```

**Figura 2 – Classificador a priori.**

Nesse estágio, cheguei a pensar em otimizar esse processo usando múltiplas threads, cheguei a implementar o mapeamento dos dados lidos. Porém analisando o tempo consumido por essa tarefa, notei que o ganho de desempenho era praticamente inexistente, uma vez que a operação realmente lenta é a leitura do arquivo de entrada.

Usando o algoritmo da Figura 2, o tempo gasto é de aproximadamente 3,2s, dos quais 1,7s são necessários para a leitura do arquivo de entrada. Usando múltiplas threads, consegui um resultado de mapeamento em 4,2s (sem dúvida pelo tempo adicional para

dividir o array de linhas lidas na entrada em subarrays menores para serem tratados em múltiplos threads, que deixou a operação mais custosa).

Uma abordagem melhor seria dividi-lo em chunks, dividindo-os entre as threads, pela complexidade deste algoritmo, optei por apenas documentá-lo neste relatório.

```
lines = open("ml-1m/ratings.dat", "r").readlines()
map(lines)

# Tentativa de usar Threads
n_threads = 10
job = len(lines) / n_threads
for i in range(0, n_threads):
    MapReduce(i, lines[job * i: job * (i + 1)]).start()
MapReduce(n_threads, lines[job * n_threads:]).start()
```

**Figura 3 – Tentativa de paralelização do processamento.**

## 3.2 Árvore de decisão

Em seguida foi desenvolvida uma implementação de árvore de decisão para o classificador baseada no algoritmo ID3. Cada nó de decisão procura escolher um atributo que maximize o ganho de informação (minimizando a entropia das decisões que ele tome).

Como destaque do algoritmo, temos a escolha do melhor atributo de decisão segundo o parâmetro que maximiza o ganho de informação, minimizando a entropia dos grupos de avaliação segundo os valores deste atributo.

```
def entropy(ratings):
    u"""Calcula a entropia de um conjunto."""
    return sum(
        (-1.0 * p / len(ratings) * math.log(1.0 * p / len(ratings)))
        for p in major_value(ratings)[0] if p != 0)


def entropy_gain(initial_entropy, ratings, attr):
    u"""Calcula o ganho de entropia para um atributo."""
    gain = initial_entropy
    for value in attributes[attr]:
        subratings = filter(lambda rate:
                            is_value(rate, attr, value), ratings)
        gain -= len(subratings) * entropy(subratings) / len(ratings)
    return gain
```

**Figura 4 – Ganho de Informação.**

O algoritmo para construção da árvore de decisão segundo o ID3 apresentado nos slides de aula é apresentado na Figura 4 abaixo:

```
def gen_tree(ratings, attributes, default):
    u"""Gera a árvore de decisões."""
    # Se não há mais avaliações
    if len(ratings) == 0:
        return Node().leaf(default, len(ratings))
    # Se avaliações são similares
    elif similar_rates(ratings):
        return Node().leaf(major_value(ratings)[1], len(ratings))
    # Se não há mais atributos de decisão
    elif len(attributes) == 0:
        return Node().leaf(mean_value(ratings), len(ratings))
    else:
        # Variável que minimiza entropia
        best = choose_attr(ratings, attributes)
        tree = Node().parent([], best)
        m = mean_value(ratings)
        # Gera subárvores de decisão
        for value in attributes[best]:
            subratings = filter(lambda rate:
                          is_value(rate, best, value), ratings)
            subattributes = {key: v for key,
                           v in attributes.items() if key != best}
            subtree = gen_tree(subratings, subattributes,
                       m).child(tree, value)
            tree.children.append(subtree)
    return tree
```

**Figura 5 – Implementação do ID3.**

A navegação nesta árvore é bem simples, basta definir um dicionário “person” com os atributos da pessoa para o qual o filme será indicado (além da árvore de decisão para o dado filme), que será realizada uma navegação pelos nós de decisão até uma folha contendo uma avaliação.

```
def navigate(node, person):
    u"""Realiza indicação de avaliação."""
    if node.rate is None:
        for child in node.children:
            if child.decision == person[node.attribute]:
                return navigate(child, person)
    return node.rate
```

**Figura 6 – Navegação na árvore de decisão.**

Encapsulando todo o código necessário para a avaliação (facilitando seu uso para um usuário), temos o método “avaliate” que executa as três etapas do classificador: gera a árvore de decisões, realiza o treinamento e então navega por ela, gerando uma classificação

sugerida. Note que a base de dados é particionada em conjunto de aprendizado (para gerar a árvore) e conjunto de treinamento (para realizar eventuais podas):

```
def avaliate(movie, person):
    u"""Realiza avaliação de um dado filme."""
    # Árvore de decisão usando todas as avaliações
    # rates = [rate for subrts in ratings.values() for rate in subrts]

    # Árvore de decisão com avaliações do filme
    database, training = ratings[movie][0::2], ratings[movie][1::2]

    # Gerando a árvore de decisões
    decision_tree = gen_tree(database, attributes, 3)
    # print_node(decision_tree, None, 0)

    # Treinando a árvore de decisões
    decision_tree = cross_tree(decision_tree, training)
    # print_node(decision_tree, None, 0)

    # Navegando na árvore de decisões
    return navigate(decision_tree, person), decision_tree
```

**Figura 7 – Avaliação da árvore de decisão.**

Uma observação muito importante é sobre os atributos que utilizei para gerar a árvore de decisão. Em vez de criar uma única árvore para todos os filmes, que certamente resulta em um número muito grande de nós, é criada apenas a subárvore referente ao filme que se deseja avaliar.

Assim, uma árvore de decisão no pior dos casos (overfitting total com árvore completa) teria aproximadamente 35 milhões de nós:

4000 filmes x 20 gêneros x 20 ocupações x 7 idades x 2 sexos = 22.4 milhões (folhas)

4000 filmes x 20 gêneros x 20 ocupações x 7 idades = 11.2 milhões

4000 filmes x 20 gêneros x 20 ocupações = 1.6 milhões

4000 filmes x 20 gêneros = 80 mil

4000 filmes = 4 mil (raiz)

Enquanto gerar apenas a subárvore relativa ao filme (descartando o atributo ID e Gênero do filme) passa a possuir no pior dos casos apenas 462 nós:

21 ocupações x 7 idades x 2 sexos = 294 nós (folhas)20

21 ocupações x 7 idades = 147 nós

21 ocupações = 21 nós (raiz)

Sem dizer que, se o atributo ID do filme possuir maior ganho de informação, ele ficará sempre perto da raiz da árvore, tendo uma árvore grande com a estrutura:

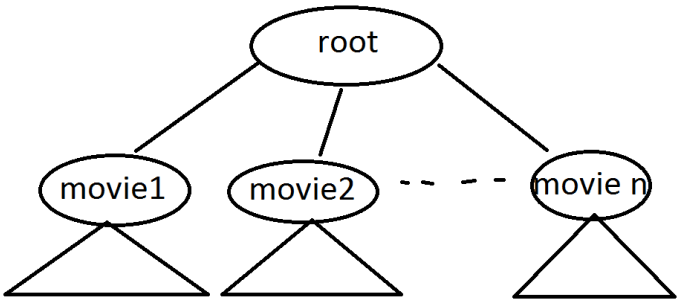


**Figura 8 – Árvore de decisão com todos os filmes.**

Enquanto a subárvore para cada filme teria uma estrutura muito similar à original, porém partindo do primeiro nível de decisão (sem contar o gênero do filme, que poderia diferenciá-la da árvore original):

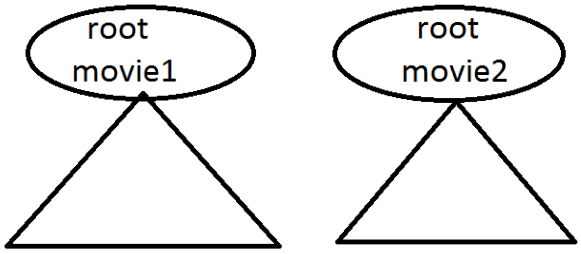


**Figura 9 – Subárvore de decisão para cada filme.**

A execução do algoritmo fica muito mais rápida no segundo caso, mantendo qualidade de resultado e ainda facilitando o processo de treinamento e manutenção da árvore de decisão. Poderíamos inclusive adotar uma política de gerar a árvore de decisão sobre demanda também, de modo a não precisar analisar avaliações de filmes pouco populares, que apenas pesariam o processamento da árvore original com todos os filmes.

```
attributes = {
    "Gender": ["M", "F"],
    "Age": ["1", "18", "25", "35", "45", "50", "56"],
    "Occupation": [str(i) for i in range(0, 21)],
    # "Genre":
    # [
    #     "Action", "Adventure", "Animation", "Children's", "Comedy",
"Crime",
    #     "Documentary", "Drama", "Fantasy", "Film-Noir", "Horror",
"Musical",
```

```
    #     "Mystery", "Romance", "Sci-Fi", "Thriller", "War", "Western"
    # ],
    # "Movie": [str(i) for i in range(1, len(movies))]
}

me = {
    "Gender": "M",
    "Age": "18",
    "Occupation": "17",
    # "Genre": movies[movie][1],
    # "Movie": movie
}
```

**Figura 10 – Atributos da árvore de decisão.**

O treinamento consiste em recursivamente testar a poda de cada um dos galhos da árvore (da raiz às folhas), verificando na base de treinamento se aquela ação melhora ou degrada os resultados. O algoritmo é detalhado abaixo:

```
def cross_val(tree, node, ratings):
    u"""Realiza validação cruzada."""
    if node.rate is None and len(ratings) > 0:
        # Faz o backup
        back_tree = copy.deepcopy(tree)
        cut_node(tree, ratings, node.number)

        # Determina qual é a melhor
        tree = should_cut(back_tree, tree, ratings)

        # Continua recursivamente
        for child in node.children:
            subratings = filter(
                lambda rate: is_value(rate, node.attribute,
child.decision),
                ratings)
            tree, node.child = cross_val(tree, child, subratings)

    return tree, node
```

**Figura 11 – Algoritmo de treinamento usando avaliação cruzada.**

Assim, foi possível encurtar ligeiramente a árvore de decisões, reduzindo o número de seus nós e melhorando a performance do classificador (evitando o problema de overfitting). Os resultados detalhados são apresentados na seção a seguir e também em anexo neste documento (árvores de decisão geradas pelo programa).

# 4. RESULTADOS

## 4.1 Implementação da análise

Nesta seção apresento os resultados encontrados decorrentes da execução do algoritmo, avaliando a consistência dos outputs gerados. Primeiro, realizei uma curta demonstração de seu funcionamento para um filme bem conhecido, analisando criticamente a árvore de decisões geradas. Em anexo temos a árvore de decisão completa gerada, com e sem o treinamento via avaliação cruzada.

```
'''
    Demonstração do programa
'''

print "Our advice: " + str(avaliate("1", me)[0])
print_node(avaliate("1", me)[1], None, 0)
analyse()
```

**Figura 12 – Curta demonstração do programa para um dado filme.**

Em um segundo momento, a performance da árvore de decisões foi comparada com um classificador a priori usando algumas propriedades bem conhecidas: taxa de acerto, matriz de confusão, erro quadrático médio e índice Kappa. A implementação destes parâmetros é descrita abaixo (de forma ineficiente, mas bastante clara para o leitor):

```
def analyse():
    u"""Analisa o classificador."""
    # Teste das minhas avaliações
    # my_rates = {"1": 5, "73": 2, "260": 4, "1210": 5, "1274": 3,
    #             "1566": 5, "1721": 2, "1907": 4, "2571": 5, "3054": 3}
    my_rates = {"2134": 3, "1265": 4, "1196": 5, "590": 4, "736": 1,
                "592": 5, "593": 4, "2088": 3, "3040": 5, "2243": 3}

    # Parâmetros de comparação
    taxa_acerto = erro_quadratico = kappa = 0
    matriz_confusao = [[0 for x in range(5)]
                       for y in range(5)]

    for movie, my_rate in my_rates.iteritems():
        avaliation = avaliate(movie, me)[0]
        # Taxa de acerto
        taxa_acerto += 1 if (avaliation == my_rate) else 0
        # Matriz confusão
        matriz_confusao[my_rate - 1][avaliation - 1] += 1
        # Erro quadrático médio
        erro_quadratico += (my_rate - avaliation)**2

    taxa_acerto = 1.0 * taxa_acerto / len(my_rates)
```

```
    esperado = 0.25
    erro_quadratico = 1.0 * erro_quadratico / len(my_rates)
    kappa = (taxa_acerto - esperado) / (1 - esperado)

    # Imprimindo resultados
    print("Taxa de acerto   : %.2f" % taxa_acerto)
    print("Matriz confusão  : " + str(matriz_confusao))
    print("Erro quadrático  : %.2f" % erro_quadratico)
    print("Índice Kappa     : %.2f" % kappa)
```

**Figura 13 – Avaliação de performance da árvore de decisão.**

Note que neste código da Figura 8 temos duas estruturas de mapa "my_rate". A primeira representa as minhas avaliações pessoais. A segunda são avaliações de um sujeito com as mesmas características que eu, extraídos da base de dados.

## 4.2 Análise de usuário do MovieLens

A seguir, vou analisar os resultados usando as avaliações deste sujeito. Sei que é um procedimento pobre, pois construí e treinei a árvore utilizando dados dele, porém acredito que ele se encaixa melhor no perfil de usuários do MovieLens do que eu (são filmes muito antigos, outra época e mentalidade). Comento melhor isso mais à frente.

Usando essa função de avaliação para o classificador a priori, obtivemos o seguinte resultado de performance. Temos uma taxa de acerto relativamente pequena, porém um erro quadrático razoavelmente pequeno, note que o índice Kappa é aproxidamente zero.

```
Taxa de acerto   : 0.30
Matriz confusão  : [[0, 0, 1, 0, 0], [0, 0, 0, 0, 0], [0, 1, 2, 0, 0], [0, 0, 2, 1, 0], [0, 0, 2, 1, 0]]
Erro quadrático  : 1.60
Índice Kappa     : 0.07
[Finished in 2.5s]
```

**Figura 14 – Resultado do classificador a priori.**

Utilizando a árvore de decisão conforme o processo descrito na seção anterior, assim, obtivemos os seguintes resultados. Note que a taxa de acerto é bem melhor, o erro quadrático é unitário (muito bom) e o índice Kappa é quase 50%, mostrando a qualidade superior de classificação da árvore de decisão.

```
Taxa de acerto   : 0.60
Matriz confusão  : [[0, 0, 1, 0, 0], [0, 0, 0, 0, 0], [0, 0, 3, 0, 0], [0, 0, 1, 2, 0], [0, 0, 1, 1, 1]]
Erro quadrático  : 1.00
```

```
Índice Kappa     : 0.47
[Finished in 4.5s]
```

**Figura 15 – Resultado da árvore sem treinamento.**

Usando parte dos dados como treinamento para o classificador (usando validação cruzada), obtemos significativa melhoria de resultado. A taxa de acerto melhorou um pouco, refletindo em queda brusca do erro quadrático médio. O índice de kappa chega a 60%, deixando o desenvolvedor orgulhoso do resultado do classificador.

```
Taxa de acerto   : 0.70
Matriz confusão  : [[0, 0, 1, 0, 0], [0, 0, 0, 0, 0], [0, 0, 3, 0, 0], [0, 0, 0, 3, 0], [0, 0, 0, 2, 1]]
Erro quadrático  : 0.60
Índice Kappa     : 0.60
[Finished in 8.7s]
```

**Figura 16 – Resultado da árvore com treinamento.**

Observe que o tempo de execução praticamente dobra quando construímos uma árvore de decisão, dobrando novamente quando um treinamento dela é exigido. Porém, o tempo continua relativamente pequeno para o processo final. Note que a classificação do classificador a priori é quase instantânea (O(1)), enquanto a navegação na árvore é bem rápida (O(n), n é a profundidade), ficando ainda mais rápida depois de treinada (chega mais rápido às folhas, em média, devido às podas realizadas).

## 4.2 Análise das minhas avaliações

Agora, a título de curiosidade, observei os resultados utilizando as minhas avaliações. Comento detalhadamente as minhas deduções sobre esse resultado curioso observado, logo a seguir:

```
Taxa de acerto   : 0.20
Matriz confusão  : [[0, 0, 0, 0, 0], [0, 0, 2, 0, 0], [1, 0, 1, 0, 0], [0, 0, 1, 1, 0], [0, 0, 1, 3, 0]]
Erro quadrático  : 1.40
Índice Kappa     : -0.07
[Finished in 2.2s]
```

**Figura 17 – Resultado do classificador a priori – minhas avaliações.**

Utilizando a árvore de decisão, ocorre melhoria significativa do desempenho do classificador, aumentando a taxa de acerto e diminuindo o erro quadrático médio. O índice Kappa também melhoria bastante:

```
Taxa de acerto   : 0.40
Matriz confusão  : [[0, 0, 0, 0, 0], [0, 1, 1, 0, 0], [1, 0, 0, 1, 0], [0, 0, 1, 1, 0], [0, 0, 1, 1, 2]]
Erro quadrático  : 1.20
Índice Kappa     : 0.20
[Finished in 4.3s]
```

**Figura 18 – Resultado da árvore de decisão inicial – minhas avaliações.**

Porém, o intrigante é notar que após realizar o treinamento da árvore (particionando o conjunto de dados usados para criar a árvore anterior), ocorre uma piora sutil do desempenho do classificador. A taxa de acerto e índice Kappa permanecem os mesmos, porém o erro quadrático médio aumenta um pouco.

```
Taxa de acerto   : 0.40
Matriz confusão  : [[0, 0, 0, 0, 0], [0, 0, 1, 1, 0], [1, 0, 1, 0, 0], [0, 0, 0, 2, 0], [0, 0, 1, 2, 1]]
Erro quadrático  : 1.50
Índice Kappa     : 0.20
[Finished in 7.3s]
```

**Figura 19 – Resultado da árvore de decisão treinada– minhas avaliações.**

Acredito que essa generalização da árvore de decisão otimize os resultados de avaliação da árvore para usuários que se encaixem no perfil da base de dados do MovieLens. Como os filmes são todos antigos e 99% desconhecidos para mim, acredito que não faço parte desse grupo, sendo natural que a generalização de indivíduos tenda a diminuir o desempenho para mim, usuário mais moderno e com opiniões ligeiramente diferentes dos outros.

Por exemplo, avaliei bem as animações da Disney (Hércules, Toy Story, Pokémon, ...) porque eu era uma criança na época de lançamento e tive uma boa experiência que carrego como bagagem de vida. No entanto, um jovem adulto engenheiro daquela época certamente não teria o mesmo tato que uma criança para gostar destes filmes, gerando uma controversa de avaliação. Generalizar os dados piora os resultados porque uma decisão mais específica e granulada é exigida neste caso.

## 5. CONCLUSÃO

Com base nesse projeto foi possível implementar uma árvore de decisão com classificação baseada em aprendizado de máquina, obtendo resultados interessantes para uma base de dados real. A etapa mais peculiar foi o treinamento do classificador, gerando uma otimização significativa para o poder de generalização, reduzindo bastante o tamanho da árvore de decisão gerada.

Como sugestão para aprimorar o classificador, poderíamos ter mais informações sobre o conteúdo do filme (produtor, elenco, tema, etc), enriquecendo a detecção de possíveis padrões relevantes para tomada de decisão. Seria interessante também agrupar os usuários por categorias conforme as suas preferências (pessoas que gostam de animações, comédias, romances, etc), que podem ser detectadas a partir do próprio conjunto de avaliações do usuário.

Como otimização deste projeto para uma aplicação real, poderíamos ter a construção prévia de todas as árvores de decisão (off-line), de modo que quando o usuário requisitasse uma classificação, seria necessário apenas carregar essa árvore em memória e então percorrê-la em O(n), em que n é a profundidade da árvore (número de atributos analisados pela árvore), que seria um procedimento muito rápido.

## 6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Slides de aula: http://www.comp.ita.br/~pauloac/

[2] Stackoverflow (sempre): http://stackoverflow.com/

[3] Wikipedia: https://www.**wikipedia**.org

## 7. ANEXO

A impressão da árvore de decisões seguiu o seguinte padrão:

*<root>, <node>, <leaf> é o tipo de nó

*[attr] é o atributo de decisão do nó

*dec é a decisão do nó em relação ao seu atributo

*rat é a avaliação sugerida pela folha

*quo é o quórum, número de rates usados para avaliar

O treinamento reduziu uma árvore de decisão de 317 linhas (ou nós) para uma menor de 65 linhas, com melhor poder de generalização ou decisão. Abaixo temos a impressão das duas, comparativamente:

Árvore de decisão após o treinamento realizado:

```
<root>
    <leaf>[Occupation]  dec: 0  rat: 4  quo: 140
    <leaf>[Occupation]  dec: 1  rat: 4  quo: 70
    <leaf>[Occupation]  dec: 2  rat: 4  quo: 49
    <node> [Occupation]     dec: 3
        <leaf>[Age]     dec: 1  rat: 4  quo: 0
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 4  quo: 4
            <leaf>[Gender]  dec: F  rat: 4  quo: 0
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 7
            <leaf>[Gender]  dec: F  rat: 4  quo: 7
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 4  quo: 2
            <leaf>[Gender]  dec: F  rat: 4  quo: 5
        <leaf>[Age]     dec: 45     rat: 4  quo: 2
        <leaf>[Age]     dec: 50     rat: 5  quo: 1
        <leaf>[Age]     dec: 56     rat: 5  quo: 1
    <leaf>[Occupation]  dec: 4  rat: 4  quo: 153
    <leaf>[Occupation]  dec: 5  rat: 4  quo: 22
    <leaf>[Occupation]  dec: 6  rat: 4  quo: 40
    <leaf>[Occupation]  dec: 7  rat: 4  quo: 104
    <leaf>[Occupation]  dec: 8  rat: 4  quo: 1
    <node> [Occupation]     dec: 9
        <leaf>[Age]     dec: 1  rat: 4  quo: 0
        <leaf>[Age]     dec: 18     rat: 4  quo: 0
        <leaf>[Age]     dec: 25     rat: 4  quo: 6
```

```
        <leaf>[Age]     dec: 35     rat: 3  quo: 7
        <leaf>[Age]     dec: 45     rat: 3  quo: 1
        <leaf>[Age]     dec: 50     rat: 5  quo: 2
        <leaf>[Age]     dec: 56     rat: 4  quo: 1
    <node> [Occupation]     dec: 10
        <node> [Gender]     dec: M
            <leaf>[Age]     dec: 1  rat: 3  quo: 24
            <leaf>[Age]     dec: 18     rat: 4  quo: 2
            <leaf>[Age]     dec: 25     rat: 4  quo: 1
            <leaf>[Age]     dec: 35     rat: 4  quo: 1
            <leaf>[Age]     dec: 45     rat: 3  quo: 0
            <leaf>[Age]     dec: 50     rat: 3  quo: 0
            <leaf>[Age]     dec: 56     rat: 3  quo: 0
        <node> [Gender]     dec: F
            <leaf>[Age]     dec: 1  rat: 4  quo: 20
            <leaf>[Age]     dec: 18     rat: 4  quo: 0
            <leaf>[Age]     dec: 25     rat: 4  quo: 0
            <leaf>[Age]     dec: 35     rat: 4  quo: 0
            <leaf>[Age]     dec: 45     rat: 4  quo: 0
            <leaf>[Age]     dec: 50     rat: 4  quo: 0
            <leaf>[Age]     dec: 56     rat: 4  quo: 0
    <leaf>[Occupation]  dec: 11     rat: 4  quo: 17
    <leaf>[Occupation]  dec: 12     rat: 4  quo: 71
    <leaf>[Occupation]  dec: 13     rat: 4  quo: 7
    <leaf>[Occupation]  dec: 14     rat: 4  quo: 60
    <leaf>[Occupation]  dec: 15     rat: 4  quo: 29
    <leaf>[Occupation]  dec: 16     rat: 4  quo: 35
    <leaf>[Occupation]  dec: 17     rat: 4  quo: 84
    <leaf>[Occupation]  dec: 18     rat: 4  quo: 7
    <node> [Occupation]     dec: 19
        <leaf>[Age]     dec: 1  rat: 3  quo: 2
        <leaf>[Age]     dec: 18     rat: 3  quo: 6
        <leaf>[Age]     dec: 25     rat: 4  quo: 3
        <leaf>[Age]     dec: 35     rat: 5  quo: 2
        <leaf>[Age]     dec: 45     rat: 4  quo: 1
        <leaf>[Age]     dec: 50     rat: 4  quo: 0
        <leaf>[Age]     dec: 56     rat: 4  quo: 0
    <leaf>[Occupation]  dec: 20     rat: 4  quo: 42
```

Árvore de decisão antes do treinamento realizado:

```
<root>
    <node> [Occupation]     dec: 0
        <node> [Age]    dec: 1
            <leaf>[Gender]  dec: M  rat: 3  quo: 7
            <leaf>[Gender]  dec: F  rat: 5  quo: 3
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 4  quo: 17
```

```
            <leaf>[Gender]  dec: F  rat: 4  quo: 10
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 3  quo: 42
            <leaf>[Gender]  dec: F  rat: 4  quo: 13
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 4  quo: 19
            <leaf>[Gender]  dec: F  rat: 4  quo: 7
        <node> [Age]    dec: 45
            <leaf>[Gender]  dec: M  rat: 4  quo: 5
            <leaf>[Gender]  dec: F  rat: 4  quo: 2
        <node> [Age]    dec: 50
            <leaf>[Gender]  dec: M  rat: 4  quo: 9
            <leaf>[Gender]  dec: F  rat: 2  quo: 2
        <node> [Age]    dec: 56
            <leaf>[Gender]  dec: M  rat: 3  quo: 4
            <leaf>[Gender]  dec: F  rat: 3  quo: 0
    <node> [Occupation]     dec: 1
        <leaf>[Age]     dec: 1  rat: 5  quo: 2
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 4  quo: 8
            <leaf>[Gender]  dec: F  rat: 4  quo: 8
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 12
            <leaf>[Gender]  dec: F  rat: 5  quo: 7
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 4  quo: 12
            <leaf>[Gender]  dec: F  rat: 5  quo: 7
        <node> [Age]    dec: 45
            <leaf>[Gender]  dec: M  rat: 4  quo: 4
            <leaf>[Gender]  dec: F  rat: 4  quo: 4
        <node> [Age]    dec: 50
            <leaf>[Gender]  dec: M  rat: 3  quo: 2
            <leaf>[Gender]  dec: F  rat: 5  quo: 1
        <node> [Age]    dec: 56
            <leaf>[Gender]  dec: M  rat: 3  quo: 3
            <leaf>[Gender]  dec: F  rat: 3  quo: 0
    <node> [Occupation]     dec: 2
        <leaf>[Age]     dec: 1  rat: 3  quo: 0
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 4  quo: 4
            <leaf>[Gender]  dec: F  rat: 3  quo: 5
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 21
            <leaf>[Gender]  dec: F  rat: 3  quo: 7
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 4  quo: 3
            <leaf>[Gender]  dec: F  rat: 4  quo: 3
        <node> [Age]    dec: 45
            <leaf>[Gender]  dec: M  rat: 4  quo: 2
            <leaf>[Gender]  dec: F  rat: 4  quo: 3
        <leaf>[Age]     dec: 50     rat: 4  quo: 1
        <leaf>[Age]     dec: 56     rat: 3  quo: 0
    <node> [Occupation]     dec: 3
        <leaf>[Age]     dec: 1  rat: 4  quo: 0
        <node> [Age]    dec: 18
```

```
            <leaf>[Gender]  dec: M  rat: 4  quo: 4
            <leaf>[Gender]  dec: F  rat: 4  quo: 0
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 7
            <leaf>[Gender]  dec: F  rat: 4  quo: 7
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 4  quo: 2
            <leaf>[Gender]  dec: F  rat: 4  quo: 5
        <leaf>[Age]     dec: 45     rat: 4  quo: 2
        <leaf>[Age]     dec: 50     rat: 5  quo: 1
        <leaf>[Age]     dec: 56     rat: 5  quo: 1
    <node> [Occupation]     dec: 4
        <leaf>[Age]     dec: 1  rat: 3  quo: 0
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 3  quo: 78
            <leaf>[Gender]  dec: F  rat: 3  quo: 36
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 30
            <leaf>[Gender]  dec: F  rat: 4  quo: 5
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 4  quo: 2
            <leaf>[Gender]  dec: F  rat: 4  quo: 2
        <leaf>[Age]     dec: 45     rat: 3  quo: 0
        <leaf>[Age]     dec: 50     rat: 3  quo: 0
        <leaf>[Age]     dec: 56     rat: 3  quo: 0
    <node> [Occupation]     dec: 5
        <leaf>[Age]     dec: 1  rat: 4  quo: 0
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 4  quo: 4
            <leaf>[Gender]  dec: F  rat: 3  quo: 1
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 10
            <leaf>[Gender]  dec: F  rat: 5  quo: 1
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 4  quo: 3
            <leaf>[Gender]  dec: F  rat: 3  quo: 2
        <leaf>[Age]     dec: 45     rat: 5  quo: 1
        <leaf>[Age]     dec: 50     rat: 4  quo: 0
        <leaf>[Age]     dec: 56     rat: 4  quo: 0
    <node> [Occupation]     dec: 6
        <leaf>[Age]     dec: 1  rat: 4  quo: 0
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 4  quo: 2
            <leaf>[Gender]  dec: F  rat: 4  quo: 0
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 15
            <leaf>[Gender]  dec: F  rat: 5  quo: 6
        <leaf>[Age]     dec: 35     rat: 5  quo: 10
        <node> [Age]    dec: 45
            <leaf>[Gender]  dec: M  rat: 4  quo: 3
            <leaf>[Gender]  dec: F  rat: 4  quo: 1
        <node> [Age]    dec: 50
            <leaf>[Gender]  dec: M  rat: 4  quo: 2
            <leaf>[Gender]  dec: F  rat: 1  quo: 1
        <leaf>[Age]     dec: 56     rat: 4  quo: 0
```

```
    <node> [Occupation]     dec: 7
        <leaf>[Age]     dec: 1  rat: 4  quo: 0
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 4  quo: 4
            <leaf>[Gender]  dec: F  rat: 4  quo: 1
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 34
            <leaf>[Gender]  dec: F  rat: 4  quo: 5
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 4  quo: 37
            <leaf>[Gender]  dec: F  rat: 5  quo: 2
        <node> [Age]    dec: 45
            <leaf>[Gender]  dec: M  rat: 4  quo: 11
            <leaf>[Gender]  dec: F  rat: 3  quo: 2
        <node> [Age]    dec: 50
            <leaf>[Gender]  dec: M  rat: 3  quo: 6
            <leaf>[Gender]  dec: F  rat: 4  quo: 1
        <leaf>[Age]     dec: 56     rat: 3  quo: 1
    <leaf>[Occupation]  dec: 8  rat: 4  quo: 1
    <node> [Occupation]     dec: 9
        <leaf>[Age]     dec: 1  rat: 4  quo: 0
        <leaf>[Age]     dec: 18     rat: 4  quo: 0
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 0
            <leaf>[Gender]  dec: F  rat: 4  quo: 6
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 4  quo: 0
            <leaf>[Gender]  dec: F  rat: 4  quo: 7
        <leaf>[Age]     dec: 45     rat: 3  quo: 1
        <leaf>[Age]     dec: 50     rat: 5  quo: 2
        <leaf>[Age]     dec: 56     rat: 4  quo: 1
    <node> [Occupation]     dec: 10
        <node> [Gender]     dec: M
            <leaf>[Age]     dec: 1  rat: 3  quo: 24
            <leaf>[Age]     dec: 18     rat: 4  quo: 2
            <leaf>[Age]     dec: 25     rat: 4  quo: 1
            <leaf>[Age]     dec: 35     rat: 4  quo: 1
            <leaf>[Age]     dec: 45     rat: 3  quo: 0
            <leaf>[Age]     dec: 50     rat: 3  quo: 0
            <leaf>[Age]     dec: 56     rat: 3  quo: 0
        <node> [Gender]     dec: F
            <leaf>[Age]     dec: 1  rat: 4  quo: 20
            <leaf>[Age]     dec: 18     rat: 4  quo: 0
            <leaf>[Age]     dec: 25     rat: 4  quo: 0
            <leaf>[Age]     dec: 35     rat: 4  quo: 0
            <leaf>[Age]     dec: 45     rat: 4  quo: 0
            <leaf>[Age]     dec: 50     rat: 4  quo: 0
            <leaf>[Age]     dec: 56     rat: 4  quo: 0
    <node> [Occupation]     dec: 11
        <leaf>[Age]     dec: 1  rat: 4  quo: 0
        <leaf>[Age]     dec: 18     rat: 3  quo: 1
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 6
            <leaf>[Gender]  dec: F  rat: 4  quo: 1
        <node> [Age]    dec: 35
```

```
            <leaf>[Gender]  dec: M  rat: 4  quo: 3
            <leaf>[Gender]  dec: F  rat: 3  quo: 1
        <node> [Age]    dec: 45
            <leaf>[Gender]  dec: M  rat: 4  quo: 4
            <leaf>[Gender]  dec: F  rat: 4  quo: 0
        <leaf>[Age]     dec: 50     rat: 3  quo: 1
        <leaf>[Age]     dec: 56     rat: 4  quo: 0
    <node> [Occupation]     dec: 12
        <leaf>[Age]     dec: 1  rat: 4  quo: 1
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 3  quo: 11
            <leaf>[Gender]  dec: F  rat: 3  quo: 1
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 42
            <leaf>[Gender]  dec: F  rat: 5  quo: 3
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 4  quo: 11
            <leaf>[Gender]  dec: F  rat: 5  quo: 2
        <leaf>[Age]     dec: 45     rat: 4  quo: 0
        <leaf>[Age]     dec: 50     rat: 4  quo: 0
        <leaf>[Age]     dec: 56     rat: 4  quo: 0
    <node> [Occupation]     dec: 13
        <node> [Gender]     dec: M
            <leaf>[Age]     dec: 1  rat: 4  quo: 0
            <leaf>[Age]     dec: 18     rat: 4  quo: 0
            <leaf>[Age]     dec: 25     rat: 4  quo: 0
            <leaf>[Age]     dec: 35     rat: 4  quo: 0
            <leaf>[Age]     dec: 45     rat: 4  quo: 0
            <leaf>[Age]     dec: 50     rat: 5  quo: 1
            <leaf>[Age]     dec: 56     rat: 3  quo: 5
        <leaf>[Gender]  dec: F  rat: 3  quo: 1
    <node> [Occupation]     dec: 14
        <leaf>[Age]     dec: 1  rat: 3  quo: 0
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 4  quo: 4
            <leaf>[Gender]  dec: F  rat: 5  quo: 2
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 3  quo: 21
            <leaf>[Gender]  dec: F  rat: 4  quo: 11
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 4  quo: 12
            <leaf>[Gender]  dec: F  rat: 3  quo: 5
        <node> [Age]    dec: 45
            <leaf>[Gender]  dec: M  rat: 4  quo: 3
            <leaf>[Gender]  dec: F  rat: 4  quo: 0
        <node> [Age]    dec: 50
            <leaf>[Gender]  dec: M  rat: 4  quo: 1
            <leaf>[Gender]  dec: F  rat: 5  quo: 1
        <leaf>[Age]     dec: 56     rat: 3  quo: 0
    <node> [Occupation]     dec: 15
        <leaf>[Age]     dec: 1  rat: 3  quo: 0
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 3  quo: 3
            <leaf>[Gender]  dec: F  rat: 5  quo: 1
        <node> [Age]    dec: 25
```

```
            <leaf>[Gender]  dec: M  rat: 4  quo: 9
            <leaf>[Gender]  dec: F  rat: 4  quo: 4
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 4  quo: 5
            <leaf>[Gender]  dec: F  rat: 2  quo: 2
        <node> [Age]    dec: 45
            <leaf>[Gender]  dec: M  rat: 4  quo: 1
            <leaf>[Gender]  dec: F  rat: 3  quo: 3
        <leaf>[Age]     dec: 50     rat: 5  quo: 1
        <leaf>[Age]     dec: 56     rat: 3  quo: 0
    <node> [Occupation]     dec: 16
        <leaf>[Age]     dec: 1  rat: 4  quo: 0
        <leaf>[Age]     dec: 18     rat: 3  quo: 1
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 11
            <leaf>[Gender]  dec: F  rat: 4  quo: 2
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 4  quo: 7
            <leaf>[Gender]  dec: F  rat: 4  quo: 3
        <node> [Age]    dec: 45
            <leaf>[Gender]  dec: M  rat: 4  quo: 2
            <leaf>[Gender]  dec: F  rat: 4  quo: 2
        <node> [Age]    dec: 50
            <leaf>[Gender]  dec: M  rat: 4  quo: 4
            <leaf>[Gender]  dec: F  rat: 3  quo: 1
        <node> [Age]    dec: 56
            <leaf>[Gender]  dec: M  rat: 4  quo: 2
            <leaf>[Gender]  dec: F  rat: 4  quo: 0
    <node> [Occupation]     dec: 17
        <leaf>[Age]     dec: 1  rat: 3  quo: 1
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 4  quo: 10
            <leaf>[Gender]  dec: F  rat: 5  quo: 1
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 24
            <leaf>[Gender]  dec: F  rat: 4  quo: 6
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 4  quo: 20
            <leaf>[Gender]  dec: F  rat: 4  quo: 5
        <node> [Age]    dec: 45
            <leaf>[Gender]  dec: M  rat: 4  quo: 8
            <leaf>[Gender]  dec: F  rat: 3  quo: 2
        <node> [Age]    dec: 50
            <leaf>[Gender]  dec: M  rat: 4  quo: 6
            <leaf>[Gender]  dec: F  rat: 4  quo: 0
        <leaf>[Age]     dec: 56     rat: 5  quo: 1
    <node> [Occupation]     dec: 18
        <leaf>[Age]     dec: 1  rat: 4  quo: 0
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 4  quo: 3
            <leaf>[Gender]  dec: F  rat: 4  quo: 0
        <leaf>[Age]     dec: 25     rat: 4  quo: 1
        <leaf>[Age]     dec: 35     rat: 4  quo: 1
        <leaf>[Age]     dec: 45     rat: 2  quo: 1
        <leaf>[Age]     dec: 50     rat: 4  quo: 1
```

```
        <leaf>[Age]     dec: 56     rat: 4  quo: 0
    <node> [Occupation]     dec: 19
        <node> [Age]    dec: 1
            <leaf>[Gender]  dec: M  rat: 3  quo: 2
            <leaf>[Gender]  dec: F  rat: 3  quo: 0
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 3  quo: 3
            <leaf>[Gender]  dec: F  rat: 4  quo: 3
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 2
            <leaf>[Gender]  dec: F  rat: 3  quo: 1
        <leaf>[Age]     dec: 35     rat: 5  quo: 2
        <leaf>[Age]     dec: 45     rat: 4  quo: 1
        <leaf>[Age]     dec: 50     rat: 4  quo: 0
        <leaf>[Age]     dec: 56     rat: 4  quo: 0
    <node> [Occupation]     dec: 20
        <leaf>[Age]     dec: 1  rat: 4  quo: 0
        <node> [Age]    dec: 18
            <leaf>[Gender]  dec: M  rat: 3  quo: 7
            <leaf>[Gender]  dec: F  rat: 4  quo: 4
        <node> [Age]    dec: 25
            <leaf>[Gender]  dec: M  rat: 4  quo: 10
            <leaf>[Gender]  dec: F  rat: 3  quo: 5
        <node> [Age]    dec: 35
            <leaf>[Gender]  dec: M  rat: 5  quo: 2
            <leaf>[Gender]  dec: F  rat: 4  quo: 5
        <node> [Age]    dec: 45
            <leaf>[Gender]  dec: M  rat: 3  quo: 2
            <leaf>[Gender]  dec: F  rat: 3  quo: 0
        <node> [Age]    dec: 50
            <leaf>[Gender]  dec: M  rat: 4  quo: 3
            <leaf>[Gender]  dec: F  rat: 4  quo: 2
        <node> [Age]    dec: 56
            <leaf>[Gender]  dec: M  rat: 4  quo: 2
            <leaf>[Gender]  dec: F  rat: 4  quo: 0
```