Disciplina: ICC204 – Aprendizagem de Máquina e Mineração de Dados 30/04/2019
Prof. Rafael Giusti (rgiusti@icomp.ufam.edu.br) atualizado em 29/05

Trabalho Prático 1

1. Descrição

Objetivo: implementar o Mini Weka, contendo seus próprios algoritmos para induzir e empregar os modelos de classificação estudados em aula. Os indutores e classificadores devem ser um ou mais programas de linha de comando.

Formato de entrada: os indutores devem ler arquivos de dados no formato ARFF ou MiniARFF, especificados neste documento. Os classificadores devem ler arquivos de dados no formato ARFF ou MiniARFF, bem como arquivos que representam os modelos induzidos, em formato especificado pelos autores.

Formato de saída: os indutores devem gerar arquivos em formato especificado pelos autores do trabalho. ~~Os classificadores devem escrever na saída padrão (`stdout`, utilizando, por exemplo, `printf()` em C ou C++, `std::cout` em C++ ou `print()` em Python ou Perl).~~ Os classificadores devem imprimir apenas a classe de cada exemplo de teste.

Restrições: não é permitido utilizar bibliotecas que implementem partes que são objeto de avaliação do trabalho. Por exemplo, você *pode* utilizar NumPy ou outra biblioteca para realizar operações de álgebra linear, mas *não pode* utilizar uma biblioteca que encontre os vizinhos mais próximos, que implemente diretamente uma função de ganho de informação ou calcule automaticamente a distância entre o exemplo de teste e os vetores de suporte. É permitido utilizar *pickles* ou outro formato automático de serialização. É permitido utilizar bibliotecas para leitura de arquivos no formato ARFF.

Formato de entrega: ~~o trabalho pode ser implementado como um único programa de linha de comando que realiza todas as tarefas ou como uma coleção de programas individuais.~~ A implementação é livre, contanto que seja bem documentada, atenda às especificações do trabalho e as dependências sejam razoáveis. Serão exigidos os códigos de todos os programas que fazem parte do trabalho, bem como um relatório descrevendo as decisões de projeto e instruções de compilação e de uso.

Linguagens: os programas podem ser implementados em C, C++, Python, Java ou Perl. É permitido utilizar bibliotecas adicionais com restrições. Em caso de dúvida, leia atentamente a especificação do trabalho. Em caso de dúvida persitente, entre em contato com o estagiário em docência da disciplina.

Prazo: o trabalho deve ser entregue até ~~05 de junho~~ 12 de junho (*deadline* estendido!).

Prorrogação: atendendo a pedidos que ainda não foram feitos (mas inevitavelmente serão), o prazo foi estendido em uma semana e serão aceitas **entregas até 18 de junho.**

Avaliação: será avaliada a corretude dos indutores e dos classificadores, bem como as decisões de projetos tomadas na implementação e na definição dos formatos de representação dos classificadores. Siga as boas práticas de programação, documente os códigos e escreva um relatório bem-feito. Qualquer extensão aos requisitos do trabalho serão levadas em conta.

Execução: os trabalhos podem ser feitos individualmente ou em grupos de até 3 alunos. Todos os alunos devem participar de todas as decisões de projeto. Poderá haver apresentações em grupo ou individuais e os alunos poderão ser chamados para defender suas contribuições. **As notas são individuais.**

2. Descrição do arquivo de entrada

Utilizaremos o formato Mini ARFF, que descreve conjuntos de dados de maneira simplificada. O formato Mini ARFF é um arquivo de texto que possui várias linhas.

É permitido utilizar também o formato ARFF. Caso decida ler arquivos em formato ARFF, o restante desta seção pode ser ignorado.

A primeira linha contém um inteiro M especificando o número de atributos. Cada uma das M linhas seguintes contém a especificação de um atributo, que pode ser numérico ou categórico. Os nomes dos atributos são sequências de 1 a 80 caracteres que podem conter apenas as letras minúsculas e maiúsculas a-z e A-Z, dígitos 0-9 e os caracteres _ e - (barra baixa ou *underscore* e hífen). Por exemplo, "_x1" é um nome de atributo válido, mas "x 1" não é. Dois atributos não podem possuir o mesmo nome.

Uma linha no formato "`num` <nome>" descreve um atributo numérico. A palavra `num` aparecerá de forma literal no arquivo Mini ARFF.

Uma linha no formato "`cat` <nome> <N> <nome$_1$> <nome$_2$> <...> <nome$_N$>" especifica um atributo categórico que pode ter N valores possíveis. Os valores permitidos para os atributos seguem as mesmas regra dos nomes dos atributos. A palavra cat aparecerá de forma literal no arquivo Mini ARFF.

A linha seguinte conterá um número inteiro K indicando o número de rótulos ou variáveis-alvos do conjunto de dados. As K linhas seguintes especificam o(s) rótulos como atributos. Neste trabalho, assuma que todos os arquivos de dados terão K=1 e o nome do atributo-alvo sempre será classe, a qual será um atributo categórico.

A linha seguinte conterá um número inteiro N, indicando o número de exemplos do conjunto de dados. As N linhas seguintes conterão os valores dos atributos para cada exemplo. Os exemplos são descritos na ordem em que os atributos foram especificados, separados por espaço ou tabulação.

O formato Mini ARFF pode permitir valores ausentes. Estes serão especificados com uma interrogação, caso sejam categóricos, ou como `NaN`, caso sejam numéricos. Tratamento de valores ausentes será considerado como "ponto extra"; a exigência mínima do trabalho é que apenas a classe poderá conter valores ausentes (no conjuto de teste).

Exemplo de arquivo Mini ARFF

```
5
num  dia
cat  aparencia   3 ensolarado    nublado chuvoso
cat  temperatura 3 quente frio   moderado
cat  umidade     2 alta   baixa
cat  vento       2 fraco  forte
1
cat classe 2 sim nao
4
1 ensolarado quente   alta   fraco   nao
2 ensolarado quente   alta   forte   nao
3 nublado    quente   alta   fraco   sim
4 chuvoso    moderado alta   fraco   sim
```

Note que a quantidade de espaços é arbitrária. Além disso, o formato Mini ARFF também pode ser utilizado para problemas de regressão. Embora eles não sejam parte deste trabalho, serão considerados como "ponto extra".

Exemplo de arquivo Mini ARFF para regressão (treinamento):

```
1
num   x
1
num   y
5
0.0   0
0.11  0.6
0.22  1.3
0.33  1.4
0.44  0.25
```

3. Descrição do arquivo de modelo

Cada indutor deverá gerar um arquivo que descreve os parâmetros do modelo induzido. O formato de arquivo é livre e deve ser especificado no relatório.

É permitido utilizar qualquer biblioteca de serialização de dados. Por exemplo, bibliotecas para leitura de JSON, serialização de objetos Python ou o formato hierárquico HDF.

Não é necessário validar os modelos gerados pelo indutor. Por exemplo, caso o arquivo `xis.json` tenha sido induzido como uma árvore de decisão, ele não será testado com o seu classificador de vizinhos mais próximos. Entretanto, esse tipo de teste é considerado uma boa prática de programação e pode contribuir positivamente na sua nota.

4. Indutores implementados

Deverão ser implementados indutores para as seguintes classes de modelos:

- Modelo probabilístico Naive Bayes (atributos categóricos e numéricos);
- Árvores de decisão (mínimo: ID3, ganho de informação e atributos categóricos);
- Regras (algoritmo de cobertura estudado em aula para atributos categóricos);
- Vizinhança (k-NN; deve ser possível utilizar distância euclidiana, Manhattan e Chebyshev);
- ~~Máquinas de vetores de suporte (mínimo kernel linear e polinomial);~~

É permitido estender qualquer um dos modelos. Por exemplo, é permitido utilizar razão de ganho na geração de árvores de decisão, mas é necessário também utilizar ganho de informação. Também é permitido acrescentar argumentos de linha de comando para especificar um conjunto de validação que permitam escolher automaticamente o melhor tamanho de vizinhança do k-NN, definir novas distâncias ou outras medidas de qualidade para indução de regras.

Todas as decisões de projeto serão levadas em conta na avaliação e devem ser documentadas no relatório. Qualquer extensão que não estiver documentada no relatório será ignorada.

Restrições adicionais podem ser impostas, desde que com bom senso e especificadas no relatório. Por exemplo, é aceitável exigir que os atributos tenham uma precisão máxima de 5 ou 6 casas decimais, mas não que os atributos numéricos sejam sempre inteiros.

5. Classificação

Uma vez induzido o modelo, deverá ser possível aplicá-lo a qualquer conjunto de testes, **mesmo que o arquivo de treinamento não esteja mais disponível**. O arquivo de testes deve ser compatível com o de treinamento. Isto é, ambos devem possuir o mesmo número de atributos, os quais devem ter os mesmos nomes e mesmos tipos.

Abaixo está um exemplo de um arquivo de teste que é **compatível** com o arquivo de treinamento mostrado no Item 2:

```
5
num  dia
cat  aparencia   3 ensolarado    nublado chuvoso
cat  temperatura 3 quente frio   moderado
cat  umidade     2 alta   baixa
cat  vento       2 fraco  forte
1
cat classe 2 sim nao
4
5 ensolarado quente   baixa   fraco   ?
6 ensolarado quente   baixa   forte   ?
7 nublado    quente   baixa   fraco   ?
8 chuvoso    moderado baixa   fraco   ?
```

Note que o arquivo de teste possui valores ausentes nas classes. O formato Mini ARFF permite valores ausentes em quaisquer atributos. Entretanto, para os fins deste trabalho, **apenas a classe** conterá valores ausentes e apenas os conjuntos de teste conterão valores ausentes. Lidar com valores ausentes em outros atributos será considerado "ponto extra".

O arquivo a seguir também é um exemplo de arquivo de teste **válido** e compatível com os dados de treino mostrados no Item 2:

```
5
num  dia
cat  umidade     2 alta   baixa
cat  vento       2 fraco  forte
cat  temperatura 3 frio    moderado   quente
cat  aparencia   3 chuvoso nublado    ensolarado
1
cat classe 2 sim nao
4
5 baixa   fraco   quente   ensolarado   ?
6 baixa   forte   quente   ensolarado   ?
7 baixa   fraco   quente   nublado      ?
8 baixa   fraco   moderado chuvoso      ?
```

Note que, nesse arquivo, houve uma mudança de ordem dos atributos e dos valores. O formato Mini ARFF **permite** que a ordem seja modificada, mas não será exigido que sua implementação seja capaz de lidar com mudança de ordem.

O arquivo na página seguinte, por outro lado, **não é compatível**, pois não possui o atributo "`dia`". Além disso, o atributo "`umidade`" possui um valor a mais e o atributo "`temperatura`" não possui o valor "`moderado`". Note que esse conjunto de testes não possui nenhum exemplo, mas é um Mini ARFF válido, pois está especificado que ele contém "zero" exemplos.

```
4
cat  aparencia   3 ensolarado    nublado chuvoso
cat  temperatura 2 quente frio
cat  umidade     3 alta   baixa  media
cat  vento       2 fraco  forte
1
cat classe 2 sim nao
0
```

Finalmente, o exemplo a seguir também **não é compatível**, pois o atributo "aparencia" possui um "A" maiúsculo, mas no arquivo de treinamento esse "A" era minúsculo. Além disso, o arquivo especifica um valor a mais para o atributo "umidade". Qualquer um desses problemas já invalidaria o arquivo, mas ele também **não é um Mini ARFF válido**, pois a *string* "média" possui um acento e a seção de dados está ausente.

```
5
num  dia
cat  Aparencia   3 ensolarado    nublado chuvoso
cat  temperatura 3 quente frio   moderado
cat  umidade     3 alta   baixa  média
cat  vento       2 fraco  forte
1
cat classe 2 sim não
← fim inesperado de arquivo!!!
```

Assuma que todos os testes serão realizados com arquivos Mini ARFF válidos, mas não necessariamente com conjuntos de treino/teste compatíveis.

6. Saída do Classificador

Para um arquivo de testes com N exemplos, o classificador deve escrever N linhas na saída padrão (isto é, `stdout`, a "tela"), em um arquivo ou no *notebook*. Cada linha deve conter a classe atribuída ao i-ésimo exemplo. **Nada mais deve ser escrito na saída padrão**. Qualquer informação adicional pode ser registrada em arquivos de *log* ou na saída de erros (`stderr`).

7. Interface, Linguagens Permitidas e Distribuição

Os trabalhos podem ser feitos nas seguintes linguagens de programação:

- C, incluindo os dialetos C89, C99 e C11;
- C++, incluindo os dialetos C++98, C++11 e C++14;
- Python 2.7 ou Python 3.x (qualquer versão do Python 3);
- Perl 5.x (qualquer versão do Perl 5);
- Java (qualquer versão).

Os programas devem utilizar **interface de linha de comando exclusivamente**. Mesmo programas feitos em Java ou Python *notebooks*. Para programas feitos em C, C++ e Java, deve ser possível compilar e executar os programas em Linux utilizando gcc, g++ ou Open JDK. Para programas em Python ou Perl, deve ser possível executar os programas em Linux utilizando o mínimo necessário de bibliotecas adicionais. Por exemplo, é permitido utilizar NumPy e Java,

mas não há necessidade de utilizar o Matplotlib. Módulos para amostragem de distribuições gaussianas podem ser utilizados, bibliotecas para ler e escrever arquivos em formato JSON, CSV, HDF ou XML são encorajados, mas bibliotecas para resolver problemas de otimização quadrática não apenas são desnecessários, mas provavelmente violam o objetivo do trabalho.

A implementação é livre, podendo ser em linha de comando, interface interativa de texto ou Jupyter Notebooks. É permitido utiizar bibliotecas para auxiliar no desenvolvimento da solução, **observando-se o bom senso**. Alguns exemplos de bibliotecas permitidas são:

- NumPy e Pandas, para ler e manipular dados matriciais;
- Biblioteca para leitura de arquivos CSV, ARFF, HDF ou XML;
- Jupyter notebooks.

Alguns exemplos de bibliotecas **não permitidas** são:

- Scikit-Learn, Keras, Weka ou qualquer implementação dos indutores e classificadores que são objeto de avaliação do trabalho;
- Bibliotecas para criação de interface gráfica ou outras bibliotecas que exigem a instalação de dependências excessivas para compilar/testar os programas.

Os trabalhos podem ser distribuídos como um único programa que realiza todas as tarefas ou como uma coleção de programas específicos para cada parte. As decisões de projeto sobre o código são livres; empregue as boas práticas, pois elas serão consideradas na avaliação.

- Caso o trabalho seja distribuído como um único programa, ele pode ser, por exemplo, um único arquivo chamado `miniweka.py` que importa módulos específicos para induzir árvores, regras etc. Esse arquivo deve receber argumentos da linha de comando que especificam se o programa está sendo utilizado para induzir ou classificar, bem como quaisquer argumentos adicionais para configurar o indutor. O classificador deve solicitar o mínimo de informação necessária ao usuário. Dados de hiperparâmetros, normalização etc. devem ser guardados no arquivo de descrição do modelo;
- Também é possível compilar diversos arquivos em C ou C++ em um único binário, chamado `miniweka`, `trabalho1` etc., que utiliza argumentos de linha de comando para realizar as tarefas. Lembre-se que os classificadores devem ser utilizáveis mesmo que os conjuntos de treinamento não estejam mais disponíveis;
- Caso o trabalho seja distribuído como vários programas, cada programa deve realizar uma parte do trabalho. Por exemplo, todos os seus arquivos `.java` podem ser compilados em um único arquivo chamado `indutores.jar` que recebe argumentos especificando qual indutor o usuário deseja empregar. Ou, como um outro exemplo, pode haver diversos arquivos com extensão `.class`, mas o arquivo `indutores.class` é o que deve ser especificado na linha de comando para induzir etc.;
- O trabalho também pode ser distribuído com vários programas ainda mais específicos. Por exemplo, pode haver um programa para induzir o modelo Naive Bayes, um programa para induzir árvores, outro para regras etc. Os mesmos programas que induzem também podem ser responsáveis por classificar seus próprios modelos ou pode haver programas separados para classificação com cada tipo de modelo.

É permitido também utilizar mais de uma linguagem de programação.

Todos os detalhes de implementação fazem parte das decisões de projeto e deverão ser registradas no relatório. Empregue boas práticas de programação.

8. Relatório

Deverá ser entregue um relatório sucinto, em formato PDF, especificando:

- As decisões de implementação dos indutores;
- Os formatos de representação dos modelos;
- Qualquer modificação realizada para estender a especificação do trabalho, tais como generalizações do formato Mini ARFF, funções de distância alternativas, funções de kernel do classificador SVM, atributos numéricos em casos que não foram solicitados, atributos categóricos no k-NN, procedimentos de validação, cálculo de acurácia etc.
- Justificativas para quaisquer restrições impostas à especificação do trabalho, que serão avaliadas pelo estagiário em docência e pelo professor. Restrições sem justificativas válidas acarretarão em perda de pontos;
- **Instruções claras de compilação e execução**. O estagiário em docência seguirá as instruções **a risca**. Trabalhos que não possam ser compilados ou executados pelo monitor por instruções pouco claras ou incorretas serão devolvidos. Se não houver tempo hábil para atualizar o relatório dentro do prazo, a nota final será **zero**.

O relatório deve conter os nomes de todos os integrantes do grupo. Todos devem trabalhar em todo o projeto. É encorajado que haja divisão de trabalho e não se espera que todos terão conhecimento absoluto dos detalhes de implementação de cada parte do trabalho, mas todos os alunos devem participar de todas as decisões de projeto e saber explicar o que foi executado.

9. Formato de Entrega

Deverá ser enviado, através do ColabWeb, um arquivo ZIP ou RAR contendo os códigos da implementação e um arquivo PDF contendo o relatório.

Não inclua arquivos binários na sua submissão, à exceção do PDF.

10. Avaliação

Os trabalhos serão corrigidos pelo estagiário em docência e revisados pelo docente. Serão avaliados:

- A corretude dos indutores e dos classificadores;
- A aderência aos requisitos do trabalho;
- A qualidade do código, que deve ser documentado e estruturado;
- As decisões de projeto devem ser bem discutidas no relatório;
- O relatório deve ser bem escrito e aderir às normas cultas da língua portuguesa.

Serão considerados adicionais:

- Programas robustos, que não falham quando mal utilizados (por exemplo, se o usuário tentar utilizar um arquivo de testes não compatível com o modelo induzido ou um classificador não compatível com o modelo induzido);
- Qualquer extensão aos itens exigidos no trabalho;

- Tratamento de valores ausentes;
- Normalização ou estandardização dos atributos. Caso seja adotada, é necessário que o usuário possa especificar para o indutor se deseja fazer uso ou não de normalização. O classificador deve normalizar ou estandardizar de acordo;
- Classes numéricas (regressão).

As notas serão atribuídas na faixa de 0 a 10. A nota 10 é reservada para trabalhos que executam perfeitamente **todos** os requisitos do trabalho. Qualquer adicional será levado em consideração na avaliação. Em caso de "*overflow*", os pontos extras poderão ser aproveitados na P1 e/ou na P2, com um limite de dois pontos por prova.

Poderá haver apresentações em grupo ou individuais e os alunos poderão ser chamados para defender suas contribuições. As notas são individuais.

11. Prazo

O trabalho deve ser entregue até 12/06, prorrogado até 18/06. Recomenda-se dividir o trabalho em etapas e concluir uma por semana.

A entrega deverá ser realizada através do ColabWeb. O espaço para submissão fechará no dia 18/06, às 23:59, seguindo o fuso horário "*anywhere on* Earth"[1]. ~~Não serão aceitas, em quaisquer hipóteses, entregas fora do prazo.~~ A entrega pode ser feita fora do prazo, via email, com dedução de 0,1 ponto por minuto depois do fechamento da atividade no ColabWeb.

1 https://www.timeanddate.com/time/zones/aoe