Descrição de trabalhos

Inteligência artificial

CONSIDERAÇÕES DE AVALIAÇÃO DOS TRABALHOS

- ▷ O trabalho deve ser feito por grupos de até 4 pessoas
- ▷ Cada trabalho valerá 20 pontos cada
- ▷ O grupo deverá obrigatóriamente estar completo na data da apresentação (alunos ausentes, independente da justificativa terá o trabalho zerado).
- ▷ O trabalho apriori terá pontuação de 100% caso rode de forma correta nos modelos fornecido pelo professor (alterações no modelo não serão aceitas) sendo os seguintes cenários
  - ▷ Cenário similar ao apresentado pelo modelo, a não execução do algoritmo neste caso terá a perda de 50% do valor da nota
  - ▷ Caso rode no primeiro cenário, será feito um cenário mais complexo sendo que a não execução nesta situação terá a perda de 10% na nota
- ▷ O grupo será arguido em relação aos resultados e implementação sendo que se um membro do grupo não der uma resposta correta ou satisfatória todos os integrantes perderão 10% da pontução.
- ▷ A versão final do agente deve ser integre impreterivelmente antes do início da aula POR EMAIL
- ▷ Trabalhos plagiados terão nota zerada.
- ▷ O algoritmo com menor custo no cenário mais complexo receberá 5 pontos extras.

---

(1) Saga das 12 casas do CDZ

Após as lutas contra os cavaleiros de prata que tentaram matar Saori Kido (a atual reencarnação de Athena) ela finalmente decide ir ao santuário destronar o grande mestre, mas, para isto é necessário que eles passem pelas 12 casa dos zodiáco protegidas pelos 12 cavaleiros de ouro (sendo estes os mais fortes).

Ao chegarem ao santuário Saori e os cavaleiros foram recepicionados por uma misteriosa figura coberta por um manto, ao chegarem ao relógio, esta figura se revelou um cavaleiro de prata, Tremy de Flecha, que atacou os cavaleiros com ilusões e cravou a flecha dourada no peito de Saori, Tremy foi morto por Seiya, mas antes de morrer disse que os cavaleiros de bronze tinham 12 horas para trazer o grande mestre para retirar a flecha, mas para isso deveriam atravessar as 12 casas do Zodíaco.

O seu objetivo é implementar um agente que seja capaz de criar uma rota que minimize o tempo gasto pelos cavaleiros para chegar na sala do grande mestre e que ao menos 1 cavaleiro chegue vivo na sala do grande mestre.

O caminho até a sala do grande mestre é constituída por uma matriz M onde cada valor $M_{i,j}$ representa um terreno, e, o custo para chegar neste terreno (em minutos) é dado a seguir:

- ▷ 0: terreno rochoso (+200 minutos)
- ▷ 1: concetro (+1 minuto)
- ▷ 2: acidentado (+3,5 minutos)
- ▷ 3: arenoso (+5 minutos)
- ▷ 10-21: Casa dos cavaleiros de ouro (Onde 10 é a casa de áries e 21 a casa de peixes)
- ▷ 22: Objetivo, sala do grande mestre

Ao chegarem na casa de um cavaleiro de um cavaleiro de ouro o custo do tempo neste caso será feito pela seguinte equação:

$$custo(X) = \frac{Poder\ cósmico\ do\ cavaleiro\ de\ ouro\ X}{\sum Poder\ cósmico\ dos\ cavaleiros\ de\ bronze}$$

Ainda, cada vez que um cavaleiro de bronze participar de uma batalha ele irá perder um ponto de vida (sendo que todos começam com 5 vidas) ao chegar em 0, o cavaleiro é considerado morto, sendo assim, deve-se chegar na sala do grande mestre um único cavaleiro ao menos com 1 ponto de vida.

Você deverá adicionar ao seu projeto o arquivo `CDZ.jar` e implementar a interface `IAgente` no qual possui os seguintes métodos

- ▷ `public Direcao mover(Jogo jogo)`: O método recebe o estado atual do jogo (tempo já gasto, posição atual dos cavaleiros...) e deve retornar a direção na qual eles devem se mover (CIMA, BAIXO, ESQUERDA, DIREITA).
- ▷ `public boolean[] batalha(Jogo jogo)`: Este método será invocado quando os cavaleiros entrarem numa batalha, neste caso, o método receberá a instância atual do jogo e deve retornar um vetor booleano com 5 valores indicando quais cavaleiros irão participar ou não da batalha.

Considerações da implementação:

- ▷ O código irá parar o agente implementado caso um movimento inválido seja executado (ex: mover para direita quando está na última coluna).
- ▷ Somente será considerado vencedor o agente passar por todas as 12 casas e ao chegar na sala do santuário ao menos 1 único cavaleiro de bronze tenha vida.
- ▷ Na execução do programa (no eclipse) você deverá escolher a classe `br.edu.ifmg.bambui.ia.cdz.Main` e passar como parâmetro o fullpath da classe do seu agente
- ▷ Já existe a classe `DummyAgent` implementada como exemplo

(2) Game of Thrones

Uma das séries atualmente mais famosas da HBO é o Game of Thrones (GoT) no qual é baseada na história fictícia da série *As crônicas de gelo e fogo* do escritor George R. R. Martin.

A história se passa numa lugar chamado de Westeros (similar à europa da idade média) onde apresenta os conflitos de diversas fámilas em busca de poder e vingança, além disso, outra ameaça iminente neste contexto é a a chegada da inverno, uma vez que a passagem de uma estação para outra pode durar anos, sendo que o inverno traz consigo um grande problema que são os outros que são pessoas que morreram e voltaram a vida sendo esta apenas uma parte de toda a trama que ocorre por trás deste cenário.

É habitual na série a morte de diversos personagens tanto por vingança quanto por "ocasionalidade". O objetivo deste trabalho é implementar 2 algoritmos de classificação (KNN, Árvore de decisão ou rede neural) com o objetivo de aprender sobre a base de conhecimento (`GoT.csv`) para predizer se uma pessoa sobrevive ou morre na série.

1. Sobre os dados

   O dataset utilizado neste trabalho é um arquivo CSV onde a primeira linha possui o cabeçalho das colunas e as linhas as ocorrências da base sendo as informações de cada coluna descritas abaixo[1]:

   a) **ID**: Código do personagem
   b) **nome**: Nome do personagem
   c) **titulo**: Título que o mesmo possui (Ex: Sir, Mão do rei, Miestre...)
   d) **masculino**: 1 para masculino, 0 para feminino
   e) **religiao**: Religião à qual o personagem segue
   f) **nascimento:** Ano de nascimento
   g) **morte**: Ano da morte (se foi relatado no livro)
   h) **casa**: Casa a qual pertence
   i) **livro1**: Se têm ponto de vista ou é mencionado no livro 1 (A guerra dos tronos)
   j) **livro2**: Se têm ponto de vista ou é mencionado no livro 2 (A fúria dos reis)
   k) **livro3**: Se têm ponto de vista ou é mencionado no livro 3 (A tormenta das espadas)
   l) **livro4**: Se têm ponto de vista ou é mencionado no livro 4 (O festim dos corvos)
   m) **livro5**: Se têm ponto de vista ou é mencionado no livro 5 (A dança dos dragões)
   n) **mae_vivo**: 1 se a mãe ainda está viva
   o) **pai_vivo**: 1 se o pai ainda está vivo
   p) **herdeiro_vivo:** 1 se o herdeiro está vivo
   q) **conjuge_vivo**: 1 se o cônjuge está vivo
   r) **casado:** 1 se for casado
   s) **nobre:** 1 se for nobre
   t) **idade:** Idade do personagem

[1] Algumas colunas possuem valor vazio (nulo) pois a informação não existe ou por não ser relevante para o personagem.

Table 1: Tabela com o resultado final da classificação

		Predito		
		Vivo	Morto	**Total**
Natural	Vivo	TP	FP	$TP + FP$
	Morto	FN	TN	$FN + TN$
	Total	$TP + FN$	$FP + TN$	$\boldsymbol{TP + FN + FP + TN}$

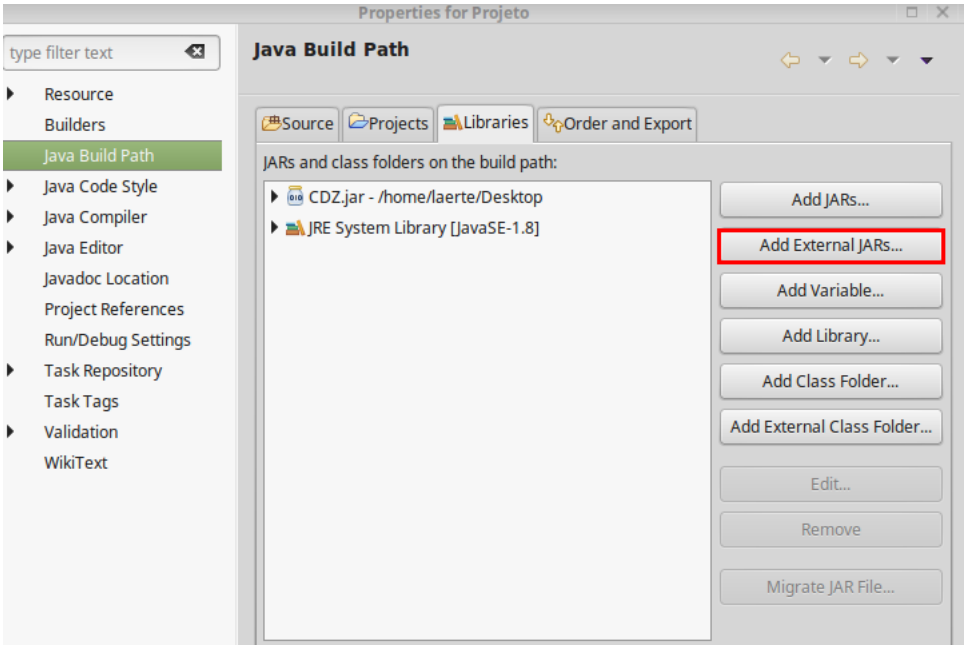
u) **mortes_correlatas**: Quantidade de mortes em que o personagem está envolvido diretamente ou indiretamente
v) **popular:** Se é considerado popular
w) **popularidade:** Índice de popularidade do personagem (0 impopular, 1 muito popular)
x) **vivo:** Se está vivo na série

2. Da predição
   a) Não se pode usar os atributos **ID**, **nome e vivo** pois todos eles são elementos discriminates na base.
   b) Deverá ser implementado uma função chamada treinar_e_testar_<Nome do algoritmo>[2] que receba como parâmetro o ID do registro na base que será excluído do treino mas o qual servirá de teste.
   c) A função deve retornar dois valores sendo estes o valor de **vivo** do registro em questão e o valor de **vivo** predito pelo classificador implementado.
3. Do resultado final e análise
   a) O resultado final deverá ser apresentado como a Tabela 1 onde cada um dos elementos significa
      A. TP: Verdadeiros positivos, na base o personagem está vivo e classificador também
      B. FP: Falso positivo, na base o personagem está vivo mas o classificador afirmou que está morto
      C. FN: Falso negativo, na base o personagem está morto mas o classificador afirmou que está vivo
      D. TN: Verdadeiro negativo, na base o personagem está morto e o classificador afirmou que está morto
   b) É necessário imprimir os valores de precisão, revocação e *f-measure* dada pelas seguintes equações respectivamente:
      A. $Prec = \frac{TP}{TP+FP}$
      B. $Recall = \frac{TP}{TP+FN}$
      C. $F = 2\left(\frac{Prec*Recall}{Prec+Recall}\right)$
   c) Para cada um dos algoritmos implementados é necessário fazer ao menos 3 variações de parâmetro (exceto para a árvore de decisão) para testes, são estes:
      A. KNN: Variar o Valor de k
      B. Rede neural: *treshold* e *learning rate*.

(3) Como importar o projeto para uso (no eclipse)

4. Crie um projeto java em branco
5. Clique com o botão direito sobre o projeto e escolha a opção Properties...
6. No menu à esquerda escolha a opção “Java Build Path”, e, na aba “libraries” clique no botão “Add external JARs”

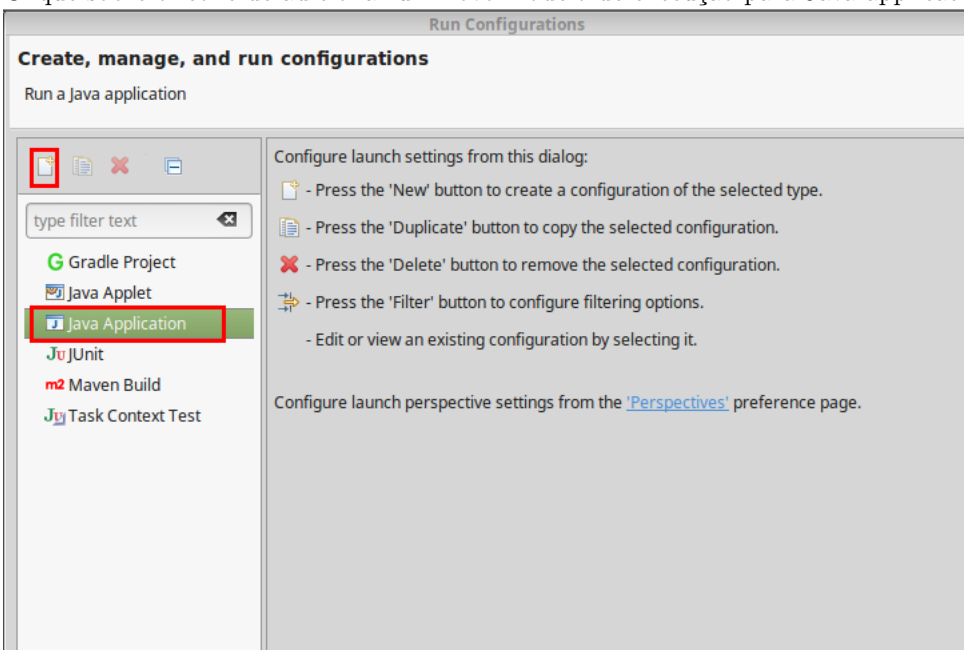
[2]Pode ser necessario outros parametros para a rotina, como por exemplo o k para a eleiço no KNN



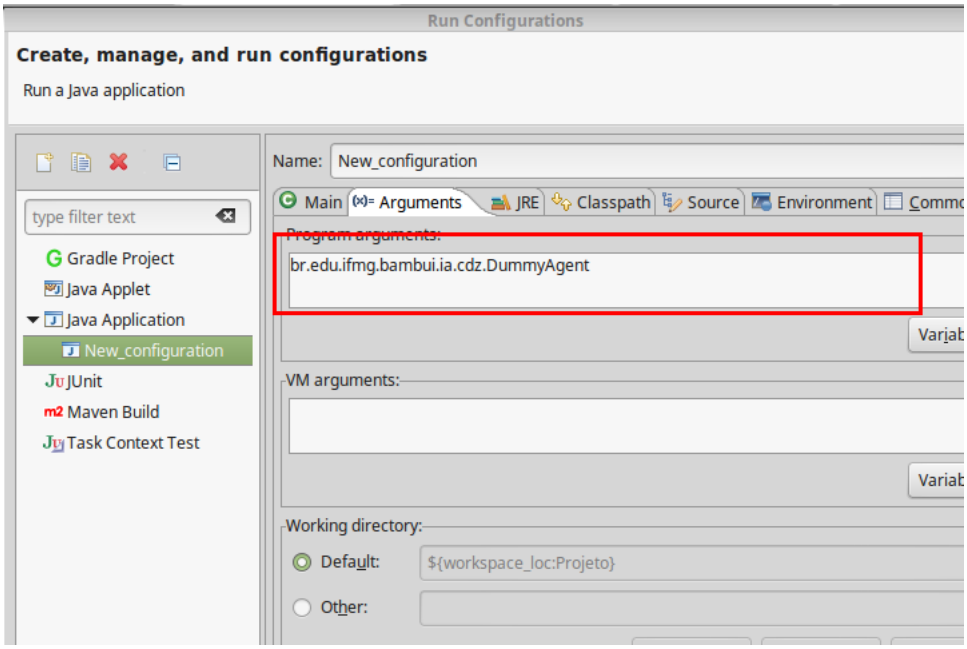
7. selecione o jar do projeto de IA

(4) Como configurar a execução do projeto (no eclipse)

8. Clique com o botão direito sobre o projeto e escolha opção "Run As" → "Run Configuration..."
9. Clique sobre o ícone de adicionar um novo modelo de execução para Java application (como na figura abaixo)



10. No fieldset "Main class:" clique sobre o botão "Search" e adicione a primeira opção (Este método main já estará incluído no pacote do projeto)
11. Para adicionar seu agente à execução na aba "Arguments" adicione o full path do agente no modelo


Run Configurations
Create, manage, and run configurations
Run a Java application
type filter text
Gradle Project
Java Applet
Java Application
New_configuration
JUnit
Maven Build
Task Context Test
Name: New_configuration
Main
Arguments
JRE
Classpath
Source
Environment
Program arguments:
br.edu.ifmg.bambui.ia.cdz.DummyAgent
VM arguments:
Working directory:
Default:
${workspace_loc:Projeto}
Other: