EDITAL 02/2026 – GABARITO

CAMPUS: Maracanã
Área de Conhecimento: Inteligência Artificial

1ª Questão:

Item (a)

Representação de conhecimento, no contexto da Inteligência Artificial simbólica, é o processo de modelar formalmente informações de um domínio por meio de estruturas explícitas — como símbolos, fatos, regras e relações — para que um sistema computacional possa armazenar esse conhecimento, raciocinar sobre ele e produzir conclusões justificáveis.

Um aspecto positivo particularmente relevante nesse tipo de abordagem é a explicabilidade. Como o conhecimento fica representado de forma explícita, torna-se possível inspecionar as regras utilizadas, rastrear as inferências realizadas e auditar a origem de uma conclusão. Em contextos como o médico, isso é importante porque a decisão sugerida pelo sistema precisa ser compreensível e justificável.

Exemplos de linguagens ou formalismos de representação de conhecimento que podem ser aceitos neste contexto incluem:

Lógica Proposicional;

Lógica de Predicados (ou Lógica de Primeira Ordem);

Lógicas Descritivas;

Regras de Produção;

Redes Semânticas;

Quadros e Roteiros (*Frames* e *Scripts*);

Lógica Nebulosa;

Teoria de Probabilidades;

Grafos de Conhecimento;

Ontologias.

Item (b)

Uma forma adequada de resolver o item é definir proposições atômicas para representar cada condição clínica relevante. Por exemplo:

F: o paciente apresenta febre;

T: o paciente apresenta tosse;

I: o paciente pode ter infecção respiratória;

A: o paciente apresenta falta de ar;

G: o caso é potencialmente grave.

A partir dessas proposições, as sentenças em linguagem natural podem ser formalizadas da seguinte maneira:

$(F \land T) \rightarrow I$

$(I \land A) \rightarrow G$

$F \land T \land A$

A conjunção "∧" representa o conectivo lógico "e", enquanto a implicação "→" representa a estrutura "se ... então ...".

Item (c)

No encadeamento para frente (forward chaining), parte-se dos fatos conhecidos e aplicam-se as regras de produção cujos antecedentes estejam satisfeitos.

Inicialmente, a base de fatos contém: F, T e A.

Aplicando a regra $(F \land T) \rightarrow I$, como F e T são verdadeiros, conclui-se I.

Com isso, a base de fatos passa a conter: F, T, A e I.

Em seguida, aplica-se a regra $(I \land A) \rightarrow G$. Como I e A são verdadeiros, conclui-se G.

Portanto, por encadeamento para frente, infere-se corretamente que o caso é potencialmente grave.

Item (d)

Encadeamento para frente e encadeamento para trás são duas estratégias clássicas de inferência em sistemas baseados em regras.

No encadeamento para frente, o raciocínio é orientado por dados (data-driven). O sistema começa com os fatos já conhecidos e vai aplicando regras para gerar novas conclusões. Essa abordagem é mais adequada quando há muitos dados disponíveis e se deseja descobrir automaticamente tudo o que pode ser inferido a partir deles, como em sistemas de monitoramento, triagem automática ou detecção de eventos.

No encadeamento para trás, o raciocínio é orientado por objetivos (goal-driven). O sistema começa com uma hipótese ou meta a ser testada e procura regras e fatos que permitam comprová-la. Essa abordagem é mais adequada quando se quer verificar uma conclusão específica, como em sistemas especialistas de consulta ou diagnóstico dirigido.

Assim, a principal diferença está no ponto de partida do processo inferencial: fatos disponíveis, no caso do forward chaining; objetivo a comprovar, no caso do backward chaining.

2ª Questão:

Item (a)

Embeddings de textos são representações vetoriais densas, em espaço numérico contínuo, usadas para codificar palavras, sentenças ou documentos de modo que relações semânticas e, em certa medida, sintáticas possam ser capturadas computacionalmente. Em vez de representar texto apenas por símbolos discretos, os embeddings projetam unidades linguísticas em vetores numéricos comparáveis entre si.

Nos embeddings estáticos, cada palavra possui um único vetor fixo independentemente do contexto em que aparece. Assim, uma palavra polissêmica, como “banco”, tende a receber a mesma representação tanto em “banco de praça” quanto em “banco financeiro”. Modelos como Word2Vec e GloVe produzem esse tipo de representação.

Nos embeddings contextualizados, a representação vetorial depende do contexto específico em que a palavra foi usada. Desse modo, a palavra “banco” pode receber vetores diferentes em contextos distintos, o que permite modelar com mais precisão ambiguidade lexical, polissemia e nuances semânticas. Modelos baseados em Transformers, como BERT, são exemplos desse tipo de abordagem.

Portanto, a diferença central é que embeddings estáticos associam um vetor único a cada termo do vocabulário, enquanto embeddings contextualizados geram vetores dependentes da sentença ou do contexto em que o termo aparece.

Item (b)

O Word2Vec foi concebido para aprender representações vetoriais de palavras a partir de padrões de coocorrência em grandes corpora textuais. Sua motivação está alinhada à hipótese distribucional: palavras que aparecem em contextos semelhantes tendem a ter significados semelhantes.

O modelo possui duas arquiteturas clássicas. Na arquitetura CBOW (Continuous Bag of Words), o objetivo é prever a palavra central a partir das palavras do contexto. Na arquitetura Skip-Gram, o objetivo é o inverso: a partir da palavra central, prever as palavras vizinhas. Em ambos os casos, a rede neural é rasa e os pesos aprendidos entre a camada de entrada e a camada oculta passam a funcionar como embeddings das palavras.

Durante o treinamento, o modelo ajusta esses pesos para maximizar a probabilidade de ocorrência de palavras em contextos coerentes. Como consequência, palavras que ocorrem em ambientes linguísticos semelhantes tendem a adquirir vetores próximos no espaço vetorial. Essa proximidade pode ser medida, por exemplo, por similaridade do cosseno.

Uma característica importante do Word2Vec é que ele não foi projetado para codificar regras semânticas manualmente, mas para induzir regularidades a partir dos dados. Por isso, relações semânticas e sintáticas emergem dos padrões de uso observados no corpus.

Item (c)

O aprendizado supervisionado em uma rede neural do tipo MLP (Multi-Layer Perceptron) com o algoritmo Back-Propagation consiste no ajuste iterativo dos pesos da rede para minimizar o erro entre a saída prevista e a saída desejada, a partir de exemplos rotulados.

Uma MLP possui:

- camada de entrada;

- uma ou mais camadas ocultas;
- camada de saída;
- pesos sinápticos e vieses.

O processo ocorre nas seguintes etapas:

1. Inicialização dos pesos

Os pesos e vieses são iniciados com pequenos valores aleatórios para permitir que os neurônios aprendam padrões distintos.

2. Forward Pass (propagação direta)

O vetor de entrada xé propagado pela rede até a camada de saída.

Para cada neurônio:

$$z = \sum w_i x_i + ba = f(z)$$

onde:

- zé a combinação linear,
- aé a ativação,
- fé a função de ativação (sigmoide, ReLU, tanh etc.).

Ao final, obtém-se a saída prevista $\hat{y}$.

3. Cálculo do erro

A saída prevista é comparada com a saída real y usando uma função de perda, como erro quadrático médio ou entropia cruzada.

Essa função mede o quanto a rede errou.

4. Back-Propagation (retropropagação)

O erro calculado na saída é propagado de trás para frente pelas camadas da rede.

Utiliza-se a regra da cadeia para calcular o gradiente da função de perda em relação a cada peso:

$$\frac{\partial L}{\partial w}$$

Isso permite identificar quanto cada peso contribuiu para o erro final.

5. Atualização dos pesos

Os pesos são ajustados usando gradiente descendente:

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

onde ηé a taxa de aprendizado.

Assim, os pesos passam a reduzir o erro nas próximas iterações.

6. Repetição

Esse processo é repetido por várias épocas até que o erro seja minimizado ou até atingir um critério de parada.

Item (d)

Redes neurais recorrentes (RNNs) e redes do tipo Transformer são arquiteturas utilizadas no processamento de textos, mas apresentam diferenças importantes em sua forma de funcionamento e desempenho.

Entre as principais diferenças, destacam-se:

1. Forma de processamento da sequência

Nas RNNs, o processamento ocorre de forma sequencial, ou seja, palavra por palavra. Cada etapa depende do estado oculto gerado na etapa anterior.

Exemplo:
para processar uma frase, a rede precisa ler os tokens em ordem, do primeiro ao último.

Nos Transformers, o processamento ocorre em paralelo. Todos os tokens podem ser analisados simultaneamente por meio do mecanismo de self-attention.

Isso torna o treinamento mais rápido e eficiente.

2. Captura de dependências de longo alcance

RNNs possuem dificuldade para capturar relações entre palavras muito distantes no texto, devido ao problema do desaparecimento ou explosão do gradiente, mesmo em variantes como LSTM e GRU.

Transformers conseguem capturar melhor essas dependências, pois cada token pode se relacionar diretamente com qualquer outro token da sequência por meio da atenção.

Isso melhora o entendimento contextual.

3. Representação de contexto

Nas RNNs, o contexto é armazenado no estado oculto, que vai sendo atualizado ao longo da sequência.

Nos Transformers, o contexto é construído dinamicamente pela atenção entre todos os tokens, sem depender de uma memória sequencial única.

Isso torna a representação mais rica e flexível.

4. Representação da ordem dos tokens

Nas RNNs, a ordem é naturalmente preservada pelo próprio processamento sequencial.

Nos Transformers, como o processamento é paralelo, é necessário utilizar Positional Encoding para informar a posição de cada token.

5. Paralelização e escalabilidade

RNNs possuem baixa paralelização, pois cada passo depende do anterior.

Transformers permitem alta paralelização e melhor aproveitamento de hardware moderno, como GPUs, sendo mais adequados para grandes volumes de dados e grandes modelos de linguagem.

3ª Questão:

(a) O **aprendizado supervisionado** o paradigma em que o modelo é treinado a partir de um conjunto de dados rotulado, ou seja, cada instancia possui um par entrada-saída (X, y) conhecido. O objetivo do modelo uma função de mapeamento $f: X \rightarrow y$, com o objetivo de generalizar o mapeamento de entradas em saídas, sendo aplicado a tarefas tais como classificação e regressão, com objetivo preditivo. Os algoritmos buscam minimizar uma função de perda que compara a saída predita com o rotulo real.
O **aprendizado não supervisionado** utiliza apenas dados de entrada não rotulados (X), buscando identificar padrões, agrupamentos intrínsecos ou estruturas latentes aos dados, sem função de perda supervisionada. como agrupamentos ou representações, com caráter exploratório. Tarefas típicas do aprendizado não supervisionado incluem agrupamento (*clustering*), redução de dimensionalidade e detecção de anomalias.

(b) O fenômeno observado é o **overfitting** (sobreajuste), caracterizado por bom desempenho no conjunto de treinamento e pior desempenho em validação, indicando que o modelo aprendeu padrões específicos e ruído dos dados de treino, sem generalizar.
Ocorre quando o modelo aprende não apenas os padrões generalizáveis dos dados de treinamento, mas também o ruído e as particularidades específicas daquela amostra. Como consequência, o modelo apresenta erro baixo no treino e erro significativamente maior na validação, situação que reflete exatamente o que os analistas mencionados no enunciado da questão observaram.

No overfitting, o modelo apresenta **baixo viés** (ele é flexível o suficiente para se ajustar aos dados de treino com precisão) e **alta variância** (pequenas mudanças nos dados de entrada produzem grandes mudanças nas predições). O gap entre desempenho de treino e validação é justamente a manifestação empírica dessa alta variância.

A questão solicita uma estratégia, e diferentes escolhas são igualmente válidas. O critério de avaliação é a coerência entre a estratégia descrita e a análise do impacto sobre viés e variância, não a estratégia em si. São exemplos aceitáveis: regularização (L1, L2, ElasticNet), early stopping, dropout, redução da dimensionalidade do modelo, entre outras.
O mecanismo comum a todas essas estratégias é restringir, de alguma forma, a capacidade do modelo de se ajustar livremente aos dados de treino, seja penalizando parâmetros de grande magnitude, interrompendo o treinamento antes da memorização, ou limitando a complexidade estrutural do modelo.

Independentemente da estratégia escolhida, o impacto sobre viés e variância segue a mesma lógica:

1. **Variância diminui:** ao restringir a capacidade do modelo, a estratégia reduz sua sensibilidade às particularidades dos dados de treino. O modelo passa a produzir predições mais estáveis diante de novas amostras, o que se traduz na redução do gap entre erro de treino e erro de validação.
2. **Viés aumenta levemente:** a mesma restrição que estabiliza o modelo também limita sua flexibilidade, podendo impedir que ele capture toda a complexidade real dos dados. Isso introduz um componente de erro sistemático — erros que persistem independentemente da amostra de treino utilizada.

O grau de intensidade da restrição governa essa escolha: restrições fracas preservam a flexibilidade do modelo (alta variância, baixo viés); restrições excessivas tornam o modelo rígido demais (baixa variância, alto viés). A escolha do ponto de equilíbrio busca minimizar o erro de generalização.

(c) No contexto do enunciado, o conjunto de dados: (1) possui atributos numéricos e categóricos; (2) apresenta alta dimensionalidade; (3) apresenta possível redundância entre atributos; e (4) possui parte dos dados rotulados e parte sem rotulação. Essas características devem orientar a organização do pipeline.

Considerações sobre as tarefas propostas para o pipeline no âmbito da questão:

1. Pré-processamento: O pré-processamento deve ocupar a primeira etapa do pipeline, pois os algoritmos dependem da qualidade e consistência dos dados. Situações tais como valores ausentes, inconsistências e atributos em escalas distintas podem comprometer as medidas de distância, decomposição matricial, otimização, treinamento dos algoritmos supervisionados e o correto funcionamento das estratégias não supervisionadas. Além disso, é comum que algoritmos demandem variáveis categóricas frequentemente precisam ser convertidas em representação numérica; e algoritmos tais como PCA, k-means, SVM e k-NN são sensíveis à escala.

2. Redução de dimensionalidade: No contexto da questão, a redução de dimensionalidade ocupa a segunda etapa do pipeline, pois, conforme descrito no contexto da questão, o conjunto apresenta alta dimensionalidade e há possível redundância entre atributos. Assim, o posicionamento da etapa de redução de dimensionalidade logo após o pré-processamento reduz redundância, minimiza ruído, reduz custo computacional, melhora a generalização e mitiga os efeitos da dimensionalidade. Além disso, algoritmos de exploração não supervisionada e modelagem supervisionada podem ser prejudicados por espaços de alta dimensionalidade.

O posicionamento da redução da dimensionalidade como terceira etapa, após a exploração não supervisionada, pode ser justificado em situações em que exploração inicial pode auxiliar entendimento estrutural ou em que for importante a descoberta preliminar de padrões. Entretanto, há de se considerar que o agrupamento em alta dimensionalidade é notoriamente problemático, a redundância afeta medidas de similaridade e a própria exploração pode se beneficiar da redução prévia. Há de se considerar, da mesma forma, que algoritmos baseados em distância podem ser prejudicados pela alta dimensionalidade e redundância explicitamente descritas no contexto do enunciado.

3. Exploração não supervisionada: A exploração não supervisionada pode revelar padrões, identificar clusters, detectar anomalias, evidenciar estruturas latentes e orientar decisões posteriores de modelagem. No contexto da questão, posicionar a exploração não supervisionada antes da modelagem supervisionada tende a ser metodologicamente mais forte, pois os padrões descobertos podem auxiliar a seleção do modelo supervisionado, a definição de hiperparâmetros e a compreensão estrutural do problema. Além disso, esta opção pode orientar a escolha do algoritmo supervisionado, de hiperparâmetros e a interpretação do problema.

O posicionamento da modelagem supervisionada antes da exploração não supervisionada faz com que, a princípio, a exploração perca parte de sua utilidade orientadora, tendo como consequência a perda de parte de papel orientador da exploração não supervisionada. Os padrões descobertos deixariam de influenciar diretamente a modelagem. Porém, tal opção justificar-se-ia em contextos específicos, tais como a

exploração posterior para análise de erros, a inspeção dos *embeddings*, a análise residual, a compreensão da estrutura aprendida e/ou o refinamento posterior do pipeline.

4. Modelagem supervisionada: A modelagem supervisionada utiliza os registros rotulados principalmente para classificação, regressão, inferência e predição. Assim, a justificativa para a aplicação deste modelo segue critérios tais como interpretabilidade, robustez, capacidade de generalização, custo computacional e comportamento frente à dimensionalidade dos dados. Esses critérios devem ser aplicados ao contexto específico do enunciado: um conjunto com atributos mistos, rotulação parcial e alta dimensionalidade. Por exemplo, algoritmos sensíveis à escala e à dimensionalidade demandam justificativa adicional quando aplicados sem redução prévia de dimensionalidade. Algoritmos robustos a essas condições dispensam essa justificativa, mas devem ter sua escolha igualmente fundamentada. Em qualquer caso, a escolha do algoritmo será avaliada pela coerência entre suas características e as do conjunto de dados descrito, não pela correspondência a uma resposta esperada.

5. Procedimento de validação: Quando considerada como macro etapa autônoma, a validação deve ocupar a última etapa do pipeline, pois depende de modelo previamente treinado, estima capacidade de generalização, auxilia na detecção de *overfitting* e permite comparação entre modelos.

O procedimento de validação pode ser considerado de forma integrada às demais etapas do pipeline — especialmente à modelagem supervisionada — desde que o candidato apresente justificativa metodológica consistente, por exemplo, validação cruzada integrada ao treinamento, seleção de hiperparâmetros ou pipelines iterativos de treinamento e avaliação.

O critério geral de avaliação do pipeline como um todo é a coerência metodológica entre as etapas: cada decisão de sequenciamento e cada escolha de técnica ou algoritmo deve ser justificada com base nas características do conjunto de dados descritas no enunciado. Respostas que proponham sequências alternativas às aqui indicadas como preferenciais serão admitidas desde que o candidato demonstre que sua escolha dialoga com essas características e que as consequências metodológicas da sequência proposta foram consideradas.

Citações não exaustivas de técnicas, algoritmos ou métricas por tarefa:

1. **Pré-processamento**: imputação de valores ausentes; padronização; normalização; *one-hot encoding*; discretização; tratamento de *outliers*; remoção de inconsistências.
2. **Redução de dimensionalidade**: PCA; seleção de atributos; LASSO; autoencoders; análise de variância; métodos baseados em importância de atributos.
3. **Exploração não supervisionada**: k-médias (*k-means*); DBSCAN; agrupamento hierárquico; *silhouette score*; análise de similaridade.
4. **Modelagem supervisionada**: regressão logística; árvore de decisão; árvores aleatórias (*random forest*); *support vector machines* (SVM); k-vizinhos mais próximos (k-NN); algoritmos supervisionados de redes neurais artificiais; XGBoost.
5. **Procedimento de validação**: *hold-out*; *k-fold cross-validation*; validação estratificada; acurácia; precisão; *recall*; F1-score; ROC-AUC; matriz de confusão.

(d) Orientações gerais de correção:

A avaliação de cada resposta deve considerar três dimensões, independentemente do algoritmo escolhido:

1. **Pseudocódigo:** deve refletir corretamente a lógica do algoritmo: suas entradas, saídas, estrutura de controle e operações principais. Não se exige sintaxe formal específica, mas a sequência lógica deve estar correta e completa.
2. **Hipóteses implícitas:** devem ser pertinentes ao algoritmo descrito e revelar compreensão dos pressupostos que sustentam seu funcionamento. A mera listagem sem conexão com o algoritmo não caracteriza resposta completa.
3. **Vantagens e limitações:** devem ser coerentes com as hipóteses apresentadas e com o comportamento conhecido do algoritmo. Respostas que listem vantagens e limitações genéricas, aplicáveis a qualquer algoritmo, sem especificidade, devem receber pontuação parcial.

Este gabarito apresentará os padrões de resposta abordados pelos candidatos: os algoritmos de classificação supervisionada k-Vizinhos Mais Próximos (k-NN), Árvore de Decisão (*Decision Tree*) e Regressão Logística; e o de agrupamento não supervisionado k-médias (*k-means*).

I. ALGORITMO: **k-NN para Classificação**

```
ENTRADA: D_treino, x_novo, k, d(.,.)

// Calcular distância de x_novo a todos os pontos de treino
PARA CADA (x_i, y_i) em D_treino FAÇA
  dist_i <- d(x_novo, x_i)
FIM_PARA

// Selecionar os k vizinhos mais próximos
k_vizinhos <- k instâncias de menor dist_i

// Retornar a classe mais frequente entre os vizinhos
RETORNAR MODA(y em k_vizinhos)
```

Hipóteses implícitas:

- Amostras próximas tendem a pertencer à mesma classe (suavidade local).
- A métrica de distância escolhida é adequada ao espaço dos dados.
- as features contribuem para a distância conforme sua escala e ponderação; na ausência de normalização ou pesos, atributos de maior escala podem dominar a distância.
- Os dados estão em escala comparável; features em escalas distintas dominam a distância.

Vantagens:

- Não paramétrico: não assume forma funcional para a fronteira de decisão
- Simples de implementar e interpretar
- Naturalmente adaptável a problemas multiclasse

- Pode modelar fronteiras não lineares.
- Não requer fase de treinamento explícita, pois o custo computacional que seria pago no treino é transferido integralmente para a inferência.

Limitações:

- Custo de inferência $O(n.p)$ por amostra, sendo n o número de instâncias de treino, e p o número de atributos, o que o torna inviável para conjuntos grandes.
- Sofre com a maldição da dimensionalidade: distâncias perdem discriminabilidade em alta dimensão
- Sensível a features irrelevantes e a diferenças de escala
- Sensível a outliers: pontos atípicos individuais podem distorcer a vizinhança e comprometer a classificação
- Requer armazenamento integral do conjunto de treino

II. ALGORITMO: **Árvore de decisão (CART – *Classification and Regression Trees*)**

```
ENTRADA: D, profundidade_max, min_amostras

FUNÇÃO CONSTRUIR(D, profundidade):
 SE critério de parada atingido ENTÃO
  RETORNAR FOLHA com classe = MODA(y em D)
 FIM_SE

 // Buscar o melhor par (feature, limiar)
 PARA CADA feature j e limiar t FAÇA
  D_esq <- {(x,y) em D : x_j <= t}
  D_dir <- {(x,y) em D : x_j > t}
  ganho <- GINI(D) - (|D_esq|/|D|)*GINI(D_esq)
                   - (|D_dir|/|D|)*GINI(D_dir)
 FIM_PARA

 Particionar D com o par (j, t) de maior ganho

 nó.esquerda <- CONSTRUIR(D_esq, profundidade + 1)
 nó.direita  <- CONSTRUIR(D_dir, profundidade + 1)
 RETORNAR nó

FUNÇÃO GINI(D):
 RETORNAR 1 - Σ_c (|{y=c em D}| / |D|)²

PREDIÇÃO para x_novo:
 Percorrer a árvore comparando x_novo ao limiar de cada nó
 RETORNAR classe da folha alcançada
```

Hipóteses implícitas:

- Splits são ortogonais às features (paralelos aos eixos).
- O melhor split global pode ser aproximado por busca gulosa local.
- A impureza de Gini, ou alternativamente entropia/ganho de informação, é proxy adequado para separabilidade.

Vantagens:

- Altamente interpretável e visualizável.
- Não requer normalização dos dados.
- Pode lidar com features numéricas e categóricas, embora algumas implementações exijam codificação.
- Captura relações não lineares e interações entre features.

Limitações:

- Alta variância: pequenas mudanças nos dados podem alterar drasticamente a estrutura da árvore.
- Tendência ao overfitting quando não há controle de profundidade.
- Fronteiras de decisão restritas a hiperplanos ortogonais aos eixos.
- Instabilidade estrutural: árvore pode mudar muito com perturbações pequenas nos dados.

III. ALGORITMO: **Regressão Logística**

```
ENTRADA: X, y, alfa, T, lambda, eps

Adicionar coluna de bias a X
Inicializar theta <- zeros

PARA t = 1 ATÉ T FAÇA
  // Calcular probabilidades preditas
  PARA CADA i FAÇA
    y_hat_i <- 1 / (1 + exp(-theta^T . x_i))
  FIM_PARA

  // Atualizar pesos via gradiente descendente
  gradiente <- (1/n) * X^T . (y_hat - y)
  theta <- theta - alfa * (gradiente + lambda * theta)

  SE ||gradiente|| < eps ENTÃO INTERROMPER FIM_SE
FIM_PARA

RETORNAR theta
```

PREDIÇÃO para x_novo:
p <- 1 / (1 + exp(-theta^T . x_novo))
RETORNAR 1 se p >= 0.5, senão 0

Hipóteses implícitas:

- Relação linear entre as features e o log-odds da classe
- Instâncias são independentes e identicamente distribuídas (i.i.d.)
- Ausência de multicolinearidade severa entre features
- A fronteira de decisão é linear no espaço original das features

Vantagens:

- Interpretável: os pesos têm significado probabilístico direto.
- Produz probabilidades calibradas, não apenas classes — isto é, o valor predito reflete adequadamente a frequência observada dos eventos.
- Eficiente computacionalmente; escala bem para grandes conjuntos.
- Regularização (L1/L2) bem estabelecida e de fácil aplicação.

Limitações:

- Inadequada para problemas com fronteiras de decisão não lineares.
- Sensível a multicolinearidade e a outliers.
- Requer features em escala comparável quando há regularização ou otimização por gradiente. Sem regularização, a escala afeta mais a otimização e a interpretação do que a validade conceitual do modelo.
- Não captura interações entre features sem engenharia manual prévia.

IV. ALGORITMO: **k-médias (*k-means*)**

ENTRADA: X, k, T, eps

// Inicializar k centroides (e.g., K-Means++)
C <- INICIALIZAR(X, k)

PARA t = 1 ATÉ T FAÇA

// Atribuir cada ponto ao centroide mais próximo
PARA CADA x_i em X FAÇA
z_i <- argmin_j ||x_i - c_j||²
FIM_PARA

```
  // Recalcular centroides como média do cluster
  PARA CADA j FAÇA
    c_j <- MÉDIA({x_i : z_i = j})
  FIM_PARA

  SE centroides não se deslocaram mais que eps ENTÃO
    INTERROMPER
  FIM_SE

FIM_PARA

RETORNAR C, Z
```

Hipóteses implícitas:

- Clusters aproximadamente globulares e relativamente bem separados.
- Clusters com tamanhos e densidades não muito discrepantes.
- O número *k* de clusters é conhecido a priori.
- A distância Euclidiana é métrica aplicada ao espaço dos dados.
- Cada ponto pertence a exatamente um cluster (particionamento rígido).

Vantagens:

- Simples, eficiente e de fácil implementação.
- Escala bem para grandes conjuntos de dados.
- Convergência garantida para um mínimo local da função objetivo.
- Interpretação direta dos centroides como representantes dos clusters.

Limitações:

- Sensível à inicialização; pode convergir para mínimos locais.
- Sensível a outliers, que distorcem os centroides.
- Não detecta clusters de forma arbitrária ou densidades diferentes.
- k deve ser especificado; escolha inadequada compromete os resultados.

4ª Questão:

a) *Um conjunto fuzzy A* definido em um universo de discurso *X* é caracterizado por uma função de pertinência μA: X → [0,1], na qual μA(x) indica o grau com que o elemento x pertence ao conjunto A.

Em conjuntos clássicos, a pertinência é binária: um elemento pertence ou não pertence ao conjunto. Na lógica fuzzy, a pertinência é gradual: μA(x)=0 indica ausência de pertinência; μA(x)=1 indica pertinência total; valores intermediários indicam pertinência parcial.

As variáveis linguísticas podem ser representadas por funções triangulares, trapezoidais ou gaussianas, com sobreposição entre os termos linguísticos. Exemplos esquemáticos aceitáveis:

Velocidade do veículo

Termo linguístico	**Interpretação esperada**
baixa	maior pertinência em velocidades pequenas
média	maior pertinência em velocidades intermediárias
alta	maior pertinência em velocidades elevadas

Fluxo de pedestres

Termo linguístico	**Interpretação esperada**
baixo	maior pertinência em fluxos pequenos
moderado	maior pertinência em fluxos intermediários
alto	maior pertinência em fluxos elevados

Risco de acidente

Termo linguístico	**Interpretação esperada**
baixo	maior pertinência para índices pequenos de risco
médio	maior pertinência para índices intermediários
alto	maior pertinência para índices elevados de risco

b) A t-norma é um operador usado para modelar o conectivo lógico fuzzy E. Ela combina dois graus de pertinência e retorna um valor no intervalo [0,1]. Exemplos corretos incluem o mínimo, T(a,b)=min(a,b), e o produto, T(a,b)=a·b.

A t-conorma é um operador usado para modelar o conectivo lógico fuzzy OU. Exemplos corretos incluem o máximo, S(a,b)=max(a,b), e a soma probabilística, S(a,b)=a+b−ab.

c) Regra fuzzy: SE velocidade do veículo é alta E fluxo de pedestres é alto ENTÃO risco de acidente é alto.

Exemplo de fuzzificação dos antecedentes: μvelocidade alta(v)=0,8 e μfluxo alto(f)=0,6. Usando a t-norma mínimo para o operador E, tem-se:

$$\mu_{antecedente} = \min(0{,}8;\ 0{,}6) = 0{,}6$$

Assim, a regra é ativada com intensidade 0,6. No modelo de inferência de Mamdani, o consequente "risco de acidente é alto" é ativado nesse mesmo grau, isto é, μrisco alto = 0,6. Esse grau pode ser interpretado como um corte ou limitação da função de pertinência do conjunto fuzzy de saída.

Caso o candidato tenha escolhido a t-norma produto, o cálculo coerente seria μantecedente = 0,8 × 0,6 = 0,48, ativando o consequente com grau 0,48.

d) Um Sistema de Inferência Fuzzy (FIS) normalmente possui os seguintes componentes: fuzzificação, base de regras, mecanismo de inferência, agregação dos consequentes e defuzzificação.

Fuzzificação: transforma entradas numéricas, como velocidade e fluxo de pedestres, em graus de pertinência nos conjuntos fuzzy.

Base de regras: contém regras linguísticas do tipo SE condição ENTÃO consequência.

Mecanismo de inferência: avalia as regras, aplica t-normas e t-conormas e determina o grau de ativação de cada regra.

Agregação: combina os consequentes fuzzy produzidos pelas diferentes regras ativadas.

Defuzzificação: converte a saída fuzzy agregada em um valor numérico único para apoiar a tomada de decisão.

A defuzzificação é necessária porque sistemas reais geralmente precisam de uma saída concreta, como um índice numérico de risco ou uma ação de controle. Métodos diferentes podem produzir decisões distintas. O centroide considera a distribuição fuzzy agregada como um todo; o método do máximo escolhe a região de maior pertinência; e a média dos máximos calcula a média dos pontos com maior ativação.

5ª Questão:

a) Crossover recombina estruturas existentes. Sua função é exploração combinatorial. Mutação: exploração aleatória.

b) Na seleção por roleta, a probabilidade de um indivíduo ser selecionado é proporcional ao seu *fitness*.

Como a *fitness é* inversamente proporcional ao custo:

$$f_i = \frac{1}{Custo_i}$$

Conclui-se que:

- Quanto menor o custo, maior será a fitness, logo maior será a chance de seleção do indivíduo.
- Maior pressão seletiva em direção a boas soluções, mas ainda mantém diversidade na população.

Número esperado de descendentes de C10

Usando a matriz:

C10=2+5+5+4+3+6+4+7+5+5=46

C10 = 46 (calculado pela matriz)

$$f_{10} = \frac{1}{46} \approx 0{,}02174$$

A probabilidade de C10 ser selecionado é

$$P(C10) = \frac{0{,}02174}{0{,}2479} \approx 0{,}0877$$

$$P(C10) = \frac{0{,}0156}{0{,}2479} \approx 8{,}77\%$$

Logo o número esperado de descendentes é:

$$E(D_{C10}) = 10 \ x\ 0{,}0877 = 0{,}877$$

Assim:

C10 gera em média $\approx 0{,}877\ descendentes$

c)

Preserva os 2 melhores indivíduos (menor custo). Garante que o melhor custo da geração não piora.

Formalmente:

$$min(P(t+1) \leq minP(t)$$

Como efeito tem-se:

- estabilidade na evolução
- aceleração da convergência
- risco de perda de soluções boas é eliminado

d)

***crossover* OX**

C1: 1-2-3-4-5-6-7-8-9-10

C5: 10-9-8-7-6-5-4-3-2-1

Máscara: 0-1-1-0-0-1-0-1-1-0

Posições com “1”: 2, 3, 6, 8, 9

Passo1:

- Copiar genes de C1 realtivo à posição dos genes 1 da máscara

_ - 2-3-_-_-6-_-8-9-_

Passo2:

Removendo {2, 3, 6, 8, 9}, restam {10, 7, 5, 4, 1} na ordem imposta por C5

Filho OX final:

10-2-3-7-5-6-4-8-9-1

***Crossover* PX**

C1: 1-2-3-4-5-6-7-8-9-10

C5: 10-9-8-7-6-5-4-3-2-1

Máscara: 1-0-0-1-0-1-1-0-0-1

- Passo1 – copiar de C1:

1-_-_-4-_-6-7-_-_10

- Passo2 – Removendo {1, 4, 6, 7, 10} de C5, restam: {9, 8, 5, 3, 2}

Filho PX final:

1-9-8-4-5-6-7-3-2-10