Técnicas e Algoritmos em Ciência de Dados
Tarefa 4

Este trabalho deve ser enviado até o dia 4 de junho de 2024, às 10h.
Envios atrasados serão penalizados em 10% por hora de atraso.

Resultados de aprendizagem avaliados

Esta tarefa oferece uma oportunidade emocionante para os alunos colocarem em prática seus conhecimentos adquiridos em sala de aula, usando árvores de decisão e florestas aleatórias para resolver um problema do mundo real na classificação e mergulhar no mundo do aprendizado não supervisionado implementando o algoritmo K-means. Os alunos também se acostumarão a gerar gráficos importantes durante o treinamento para analisar o comportamento dos modelos.

Instruções

Identificador
Por favor, use o **mesmo número aleatório de 6 dígitos que você usou para as TAREFAS 1,2,3** e escreva-o na primeira célula do notebook. Certifique-se de manter uma cópia desse número, pois ele será usado para fornecer o feedback.

Submissão
Envie seus arquivos através do ECLASS. Os arquivos que você envia não podem ser substituídos por mais ninguém e não podem ser lidos por nenhum outro aluno. No entanto, você pode substituir seu envio quantas vezes quiser, reenviando, embora apenas a última versão enviada seja mantida.

Se você tiver problemas, no último minuto, envie sua tarefa por e-mail como um anexo em alberto.paccanaro@fgv.br com o assunto "URGENTE – SUBMISSÃO TAREFA 4". No corpo da mensagem, explique o motivo de não enviar através do ECLASS.

IMPORTANTE

- Seu envio consistirá em um único notebook do Python implementando suas soluções.
- **O nome do arquivo será o número aleatório que o identifica (por exemplo, 568423.ipynb)**
- Esta tarefa consiste em 2 partes. Certifique-se de que o código de ambas as partes é colocado nas células de código relevantes no notebook.
- NÃO ENVIE NENHUM CONJUNTO DE DADOS, apenas o código.
- Qualquer função auxiliar que você usará deve ser incluída no notebook – não envie scripts adicionais.

Todo o trabalho que você enviar deve ser exclusivamente seu próprio trabalho. As submissões de trabalhos do curso serão verificadas para isso.

Critérios de marcação

Este trabalho de curso é avaliado e obrigatório e vale 10% da sua nota final total para este curso. Para obter nota máxima para cada pergunta, você deve respondê-la corretamente, mas também completamente. Serão dadas notas para escrever código de estrutura de poço.

IMPORTANTE: Em geral, você pode usar *numpy* ou outras bibliotecas básicas para operações de matriz, mas você não pode usar qualquer função de biblioteca que implementaria alguns dos algoritmos que você é obrigado a implementar. Se você está em dúvida sobre uma função específica, envie-nos um e-mail.
Além disso, para o ajuste de parâmetros, você não tem permissão para usar qualquer função de pesquisa de "grid", como GridSearchCV do sklearn ou funções equivalentes.

TAREFA IV

Parte 1 – Árvores de Decisão para Classificação (valor: 30%)

Neste exercício, você implementará uma árvore de decisão para classificar se dois medicamentos terão uma interação adversa com base em um subconjunto do conjunto de dados de interações medicamento-medicamento (DDIs) do Twosides. Você usará o ganho de informação baseado na entropia como medida de impureza como critério de divisão. A profundidade máxima e um número fixo mínimo de instâncias por folha serão seus critérios de parada. O conjunto de dados contém combinações de 40 medicamentos (formando 780 pares), com cada par representado por uma combinação da representação PCA dos alvos proteicos de cada medicamento individualmente.

Para concluir este exercício, você escreverá código para criar uma árvore de decisão para esse problema:

1. **Divisão de conjunto de dados:**
    a. Carregue o conjunto de dados fornecido em seu código.
    b. Divida o conjunto de dados em treinamento (90%) e teste (10%). Você não precisará de um conjunto de validação porque no ponto 4 você implementará a validação cruzada.
2. **Implemente uma função para aprender Árvores de Decisão – os principais passos conceituais são detalhados abaixo:**
    a. Inicialize uma árvore de decisão vazia.
    b. Implemente uma função recursiva para criar a árvore de decisão:
        i. As condições de parada para a função recursiva são [nota: satisfazer apenas uma delas é suficiente para parar a recursão]:
            - ✓ Se a profundidade máxima for atingida, pare de crescer a árvore e crie um nó de folha com a frequência da classe positiva para as instâncias restantes.
            - ✓ Se o número de instâncias em um nó folha for menor que o número mínimo de instâncias por folha, crie um nó folha com a frequência da classe positiva para essas instâncias.
        ii. Seu código calculará o Ganho de Informações (baseado na Entropia) para cada valor possível de cada atributo e escolherá o atributo e o valor que maximiza o Ganho de Informação (explicação abaixo).
        iii. Seu código criará um novo nó interno usando o atributo e o valor escolhidos.
        iv. Seu código chamará recursivamente a função de compilação em cada subconjunto de instâncias criadas pela divisão.
3. **Implementar uma função de classificação.** Implemente uma função para classificar novas instâncias usando a árvore de decisão:

a. Para cada instância, percorra a árvore de decisão comparando seus valores de atributo com os nós de decisão e mova a árvore para baixo com base nos valores de atributo até que um nó de folha seja alcançado.
b. Retorne a frequência da classe positiva associada ao nó da folha como a previsão para a instância.

4. **Execute seu algoritmo e avalie seu desempenho:**
   - **a.** Chame a função de construção com o conjunto de treinamento para construir a árvore de decisão. Você variará a profundidade máxima e usará a 3-fold cross-validation (validação cruzada) para encontrar os parâmetros ótimos. Os valores a serem testados para a profundidade máxima são: 4, 6 e 8. Você fixará o número mínimo de instâncias por folha em 15.
     - *i. Observação: o treinamento pode levar cerca de 10 minutos (ou mais, dependendo da sua implementação). Isso não é um problema e pontos não serão deduzidos.*
   - **b.** Relate a acurácia média (limiar: 0,5) e AUROC para a melhor profundidade máxima da validação cruzada.
   - **c.** Treine uma árvore de decisão com os melhores parâmetros usando todo o conjunto de treinamento, e calcule a acurácia e a AUROC no conjunto de teste.

Para selecionar a melhor divisão em cada nó, você usará o você usará o ganho de informação baseado na entropia. A entropia mede a impureza de um nó em uma árvore de decisão. Para calcular o ganho de informação baseado na entropia, siga estes passos:

- Para cada divisão potencial (atributo e valor):
  a. Calcule a entropia para o nó *m* (antes de qualquer divisão) usando a distribuição de classes dentro do nó, usando a seguinte fórmula:
  $H_m = -\sum_{k=1}^{K} \hat{p}_{mk} \log_2(\hat{p}_{mk})$, onde $\hat{p}_{mk}$ representa a proporção de instâncias no nó m que pertencem à classe k.
  b. Calcule a entropia para cada resultado possível. Isso envolve as seguintes etapas:
     i. Divida os dados com base nos possíveis resultados do atributo.
     ii. Calcule a entropia para cada subconjunto resultante usando a mesma fórmula da etapa a.
  c. Calcule a entropia ponderada somando as entropias de cada subconjunto, ponderados pela proporção de instâncias que ele representa no nó original. A fórmula para a entropia ponderada (W) é a seguinte:
  $W_V = \sum_{v}^{V} \frac{|S_v|}{|S|} H_{S_v}$ onde S_v é o nó após a divisão e a soma itera sobre todos os nós filhos; $|S_v|$ representa a cardinalidade do nó e $|S|$ a cardinalidade do nó antes da divisão; G_{S_v} representa a entropia do nó.
  d. Calcule o ganho de informação subtraindo a entropia ponderada obtida na etapa c. da entropia do nó atual. A fórmula é a seguinte:

$$InformationGain = H_{node} - W_V$$

Parte 2 – Floresta Aleatória para Redes de Classificação (valor: 30%)

Neste exercício, você expandirá o exercício anterior e implementará Florestas Aleatórias. Você construirá um conjunto de árvores de decisão e as usará para a mesma tarefa de classificação da Parte 1. O conjunto de dados utilizado para este exercício será o mesmo do exercício anterior. Sua tarefa é escrever código para construir um modelo de Floresta Aleatória, avaliar seu desempenho e compará-lo com a implementação da árvore de decisão.

Para concluir este exercício, você escreverá código para implementar a floresta aleatória para esse problema:

1. **Divisão do Conjunto de Dados:** para este exercício, você usará 80% para treinamento, 10% para validação e 10% para teste.
2. **Implemente uma função para aprender Random Forest – as principais etapas são detalhadas abaixo:**
    a. Inicialize uma floresta aleatória vazia.
    b. Determine o número de árvores de decisão que vão ser incluídas na floresta (por exemplo, 15) e o número de atributos aleatórios que vão ser considerados, geralmente num_features $\approx \sqrt{p}$ onde p e o número total de atributos.
    c. Implemente um loop para criar o número especificado de árvores de decisão:
        i. Gere uma amostra de bootstrap a partir do conjunto de treinamento (amostragem com substituição).
        ii. Crie uma árvore de decisão usando o exemplo de bootstrap, usando sua implementação da Parte I.
        iii. Adicione a árvore de decisão construída à Floresta Aleatória.
3. **Implementar uma função de classificação.** Implemente uma função para classificar novas instâncias usando a Floresta Aleatória:
    a. Para cada instância, passe-a por todas as árvores de decisão na Floresta Aleatória e colete as previsões. Observe que você deve binarizar a previsão de cada árvore de decisão, ou seja, usar um limiar de 0.5 para determinar o rótulo de classe real.
    b. A previsão para a floresta aleatória será a frequência da classe positiva nas previsões coletadas por todas as árvores de decisão.
4. **Execute seu algoritmo e avalie seu desempenho:**
    a. Chame a função para aprender a Floresta Aleatória com seu conjunto de treinamento. Você variará os diferentes parâmetros da Floresta Aleatória para observar seu efeito no desempenho no conjunto de validação. Você usará o conjunto de treinamento para aprender a árvore e o conjunto de validação usando a Área sob a Curva Roc (AUROC) para encontrar os parâmetros ideais. Novamente, mantenha árvores de profundidade não superior a 10 e min_instances não menores que 15, e não construa muitas árvores de decisão, pois isso pode atrasar bastante o tempo de treinamento.
        i. Não exceda 25 ou 30 árvores de decisão.

ii. Teste apenas algumas combinações de parâmetros, em torno de 3 a 5.

b. Crie uma Floresta Aleatória usando os conjuntos de treinamento + validação com a melhor combinação de parâmetros.
c. Classifique as instâncias do conjunto de testes usando a Floresta Aleatória, calcule a acurácia (limiar 0.5) e a Área sob a Curva ROC (AUROC) e relate os resultados.

5. **Explore:** compare o desempenho da sua implementação de árvores de decisão com florestas aleatórias. Lembre-se de que, para esta etapa, use a divisão de dados que você utilizou na parte II.

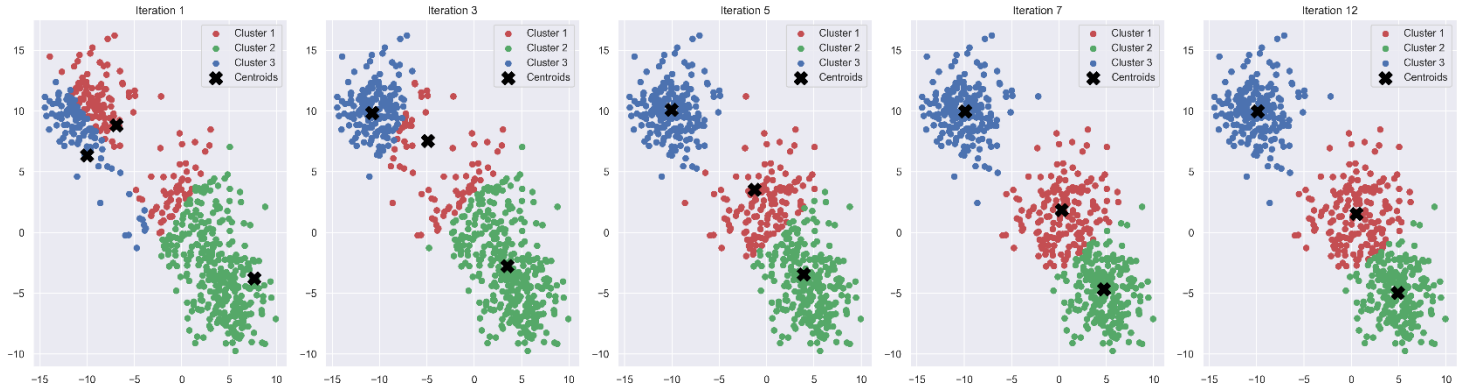
Parte 3 – Agrupamento com K-means (valor: 40%)

Neste exercício, você explorará o clustering implementando o algoritmo K-means. Você escreverá código para executar o agrupamento K-means enquanto visualiza o movimento dos centroides em cada iteração.

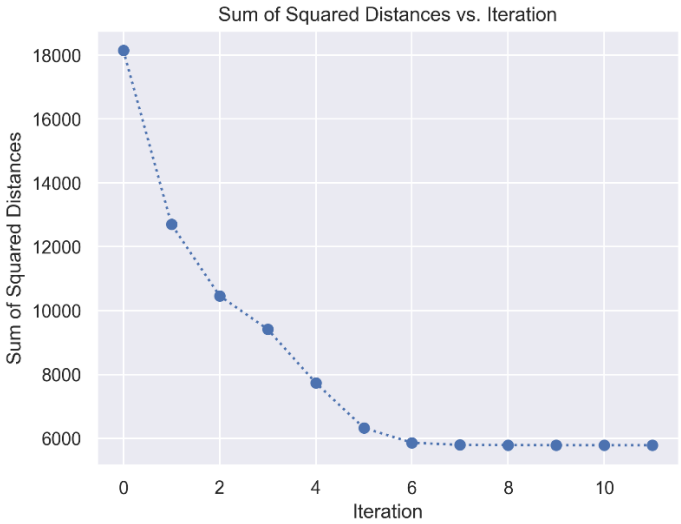
Para concluir este exercício, você escreverá código para implementar K-means para clustering:

1. **Preparação do conjunto de dados:** execute as células fornecidas no bloco de anotações que geram os pontos de dados artificiais para este exercício.
2. **K-means Clustering:**
   a. Inicialize centroides de cluster K selecionando pontos K do conjunto de dados aleatoriamente.
   b. Implemente um loop para executar as seguintes etapas até a convergência (ou até que um número máximo especificado de iterações seja atingido, por exemplo, 150):
      i. Atribua cada ponto de dados ao centroide mais próximo (você terá que calcular a distância euclidiana entre o ponto de dados e cada centroide).
      ii. Atualize cada centroide movendo-o para a média de todos os pontos de dados atribuídos a ele.
      iii. Verifique a convergência comparando os novos centroides com os centroides anteriores. Se a diferença for menor que um $\epsilon = 1e - 4$, saia do loop.
3. **Visualização do movimento centroide:**
   a. Em 5 momentos diferentes durante o treinamento, plote uma figura mostrando os centroides e os pontos. A Figura 1 deve mostrar a situação no início, antes do aprendizado. A Figura 5 deve mostrar a situação ao final do aprendizado. As Figuras 2 a 4 restantes devem mostrar situações intermediárias (veja um exemplo abaixo e observe que os pontos de dados usados aqui são diferentes dos que você usará, estes foram gerados para fins de ilustração).
   b. Para cada figura, cada centroide será representado por uma grande cruz preta e cada cluster com uma cor diferente, os pontos devem ser coloridos de acordo com seu respectivo agrupamento.

4. **Soma das distâncias quadradas:**
   a. Juntamente com a plotagem do movimento do centroide, calcule a soma das distâncias ao quadrado em cada iteração da seguinte maneira: $\sum_{j=1}^{K}\sum_{n\in S_j} d\left(x_n, \mu_j\right)^2$, onde K é o número de clusters, x_n representa o ponto n^{th} de dados, $n \in S_j$ indica um conjunto de pontos que pertencem ao cluster S_j, μ_j é a média dos pontos de dados em S_j e $d\left(x_n, \mu_j\right)$indica a distância euclidiana entre x_n e μ_j.
   b. Faça um gráfico da soma das distâncias ao quadrado em cada iteração (veja um exemplo abaixo, novamente, observe que os dados usados para esse gráfico são diferentes dos seus).



Exemplo da Visualização do Movimento Centroide.



Exemplo da Soma de Distâncias Quadradas entre iterações.