Técnicas e Algoritmos em Ciência de Dados
Tarefa 4

Este trabalho deve ser enviado até o dia 15 de junho de 2023, 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
Use o **número aleatório de 6 dígitos que você usou para o primeiro trabalho do curso** 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 enviados não podem ser substituídos por ninguém e não podem ser lidos por nenhum outro aluno. Você pode, no entanto, substituir seu envio quantas vezes quiser, reenviando, embora apenas a última versão enviada seja mantida.

Se você tiver problemas na última hora, envie um e-mail como anexo para alberto.paccanaro@fgv.br com o assunto "URGENTE – ENVIO DE TAREFA 4 ". NO corpo da mensagem, explique o motivo de não enviar através do ECLASS.

IMPORTANTE

- Seu envio consistirá em um único bloco de anotações Python implementando suas soluções.
- **O nome do arquivo será o número aleatório que o identifica (por exemplo, 568423.ipynb)**
- Este curso é composto por 2 partes. Certifique-se de que o código de ambas as partes está 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

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

Neste exercício, você implementará uma árvore de decisão para classificar se a renda de uma pessoa excede US $ 50 mil / ano com base em dados do censo (adult_census_subset.csv no ECLASS). Você usará o Ganho de Informação com base no Índice de Gini como a medida de impureza como critério de divisão. A profundidade máxima e o número mínimo de instâncias por folha serão seus critérios de parada. Esteja ciente de que algumas das variáveis neste conjunto de dados são nominais (ou categóricas).

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 três conjuntos: treinamento, validação e teste, com uma proporção de 70/15/15, respectivamente.
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 (com base no Índice de Gini) 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 que usa o conjunto de treinamento para construir a árvore de decisão. Você variará a profundidade máxima e o número mínimo de instâncias por folha para observar seus efeitos no desempenho da árvore de decisão. 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.
    Tente apenas árvores de profundidade não superior a 10 e min_instances não menores que 10. Se você tentar valores mais extremos, o tempo de treinamento pode ser demais.
    b. Construa uma árvore de decisão usando os conjuntos de treinamento + validação com a melhor combinação de parâmetros.
    c. Calcule a acurácia (limiar 0.5) e AUROC da árvore de decisão no conjunto de testes e relate-as.

Para selecionar a melhor divisão em cada nó, você usará o Ganho de Informação com base no Índice de Gini. O Índice de Gini mede a impureza de um nó em uma árvore de decisão. Para calcular o Ganho de Informação com base no Índice de Gini, siga estes passos [nota: o mesmo é explicado nos slides para o caso de entropia]:

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

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

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 de Conjunto de Dados:** use as mesmas divisões que você usou para a Parte 1.
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, 20) 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 10, e não construa muitas árvores de decisão, pois isso pode atrasar bastante o tempo de treinamento.

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. **Experimentação:** Compare o desempenho de Florestas Aleatórias com a implementação de árvore de decisão única do exercício anterior, relatando o desempenho no conjunto de testes em uma tabela (um dataframe ou o exemplo de markdown fornecido no Laboratório 12).

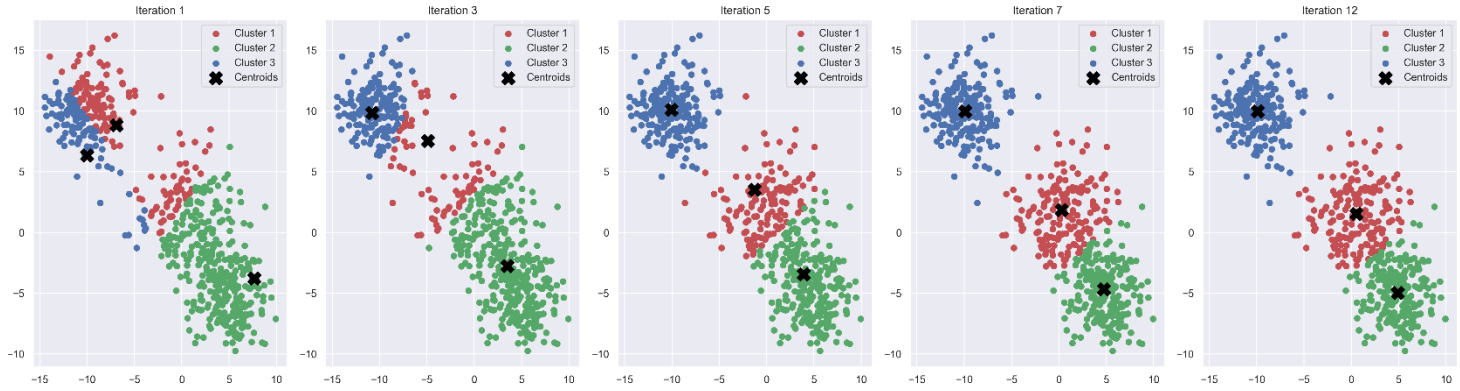
Parte 3 – Agrupamento com K-means (valor: 40%)

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.

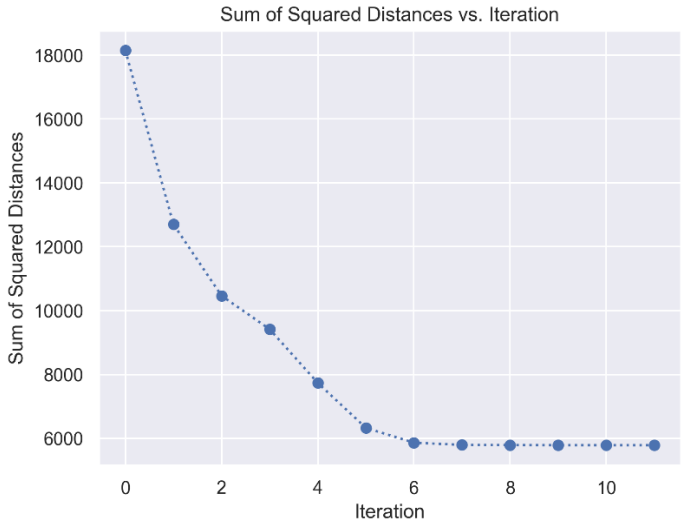
Para concluir este exercício, você escreverá código para implementar K-means para clustering:

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.

Pontos de bônus: Implemente o algoritmo para cada exercício como uma classe para obter pontos de bônus. Nesta tarefa você pode obter até 15% de pontos de bônus, 5% para cada exercício.