UNIVERSIDADE FEDERAL
DO RIO DE JANEIRO

Centro de Ciências Matemáticas e da Natureza
Instituto de Computação
Trabalho final – Introdução à programação C/C++

Professor: Giomar Sequeiros
Período: 2023 – I

Instruções:
- O projeto pode ser desenvolvido em grupos de até 5 alunos
- O **código** deve estar devidamente **comentado** e documentado.
- Fazer upload do código fonte + relatório final exclusivamente pelo seguinte formulário: https://forms.gle/1JT7vVKExGtYTPBH6

Cronograma de envio:
- Apresentação grupal (no laboratório) até **06/07/2023**
- Envio da versão final do código fonte e relatório até **10/07/2023**

Pontuação: 20% da média final

Aplicação de aprendizado máquina (*machine learning*)

ENUNCIADO

Criar um programa que permita analisar **o banco de dados Iris** e permita criar um **modelo** de **aprendizado máquina** capaz **caracterizar** uma flor Iris dadas as medidas das pétalas e sépalas.

O banco de dados está disponível em um arquivo de texto **iris.csv** (formato csv - valores separados com vírgulas) disponível no link a seguir:
https://drive.google.com/file/d/1dGHLZoD5-4cnKv2F_tO7Zpw5DxmlA6Ss/view?usp=share_link

Este banco de dados contém um conjunto de 151 linhas com cinco colunas (atributos) listados a seguir:
- Comprimento da sépala
- Largura da sépala,
- Comprimento da pétala,
- Largura da pétala
- Tipo

A continuação mostra-se as 4 primeiras linhas do arquivo **iris.csv**

```
comprimento_sepala,largura_sepala,comprimento_petala,largura_petala,tipo
5.1,3.5,1.4,0.2,setosa
4.9,3.0,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
```

A primeira linha contém apenas os nomes das colunas. A partir da segunda linha são armazenados 5 valores (separados por vírgula). Os quatro primeiros valores são numéricos e representam as dimensões de uma flor íris (em centímetros), o último valor é categórico pudendo ser: setosa, versicolor ou virginica. Ao todo são 150 registros de medições com 50 registros para cada tipo de flor de íris (veja figura abaixo)

Quando o **programa** for **inicializado**, o arquivo iris.csv deve ser lido e **armazenado** em **arrays**, os quais devem estar **disponíveis** enquanto o programa estiver em **execução**. Estes arrays devem ser nomeados e estruturados da seguinte forma:

- **atributos**: armazena os nomes das colunas, array de strings de **1x5**
- **caracteristicas:** matriz que armazena as medidas (valores numéricos) das flores, array de float de **150x4**
- **classes**: armazena o tipo de flor associado a cada medida, array de string de **150x1**

Será necessário criar uma função que receba o caminho do arquivo e crie os arrays citados acima. **Dica**: utilize a função **strtok** para separar os valores de uma linha de arquivo usando delimitador vírgula.

Após o programa carregar satisfatoriamente os dados, deve exibir um **menu** com as seguintes **opções**:

[1] – Mostrar estatísticas
[2] – Classificar amostra
[3] – Sair

A **opção [1]** deve apresentar um **resumo** das **estatísticas** do banco de dados similar ao mostrado abaixo:

```
Resumo base de dados iris: 150 elementos
   comprimento_sepala:
      Mínimo: 4.3
      Máximo: 7.9
      Média: 5.843
      Desvio padrão: 0.828
   largura_sépala:
      .......
   comprimento_pétala:
      .......
   largura_pélata:
      .......
   tipo:
      Setosa: 50 ocorrências
      Virgínica: 50 ocorrências
      Versicolor: 50 ocorrências
```

Ou seja, como as medidas das flores estão armazenados na matriz **caracteristicas**, criar funções que recebam uma matriz e o índice da coluna e retorne a respectiva estatística. Por exemplo:

```
float minimo(float caracteristicas[LINHAS][COLUNAS], int col)
```

A chamada **minimo(caracteristicas, 0)** retorna o valor mínimo da matriz **caracteristicas** correspondente à coluna **0** (Comprimento da sépala)

A **opção [2]** deve permitir fazer uma classificação automática, ou seja dada as medidas de uma flor o programa deve identificar o tipo ao qual a amostra pertence. Para isso, o programa deve solicitar que o usuário digite 4 medidas em centímetros e armazená-los em um array. A seguir o programa deve fazer uma busca por semelhança na matriz de características. Esta busca deve seguir a ideia do algoritmo **KNN** (k-nearest neighbors ou k-ésimo vizinho mais próximo) quando o valor de k=1, ou seja, deve procurar pelo elemento mais similar na matriz de caracteristicas e retornar a classe que esse elemento possui.

Considere a função:

```
int classificar_1NN(float caracteristicas[LINHAS][COLUNAS], float amostra)
```

A função classificar_1NN recebe a matriz de características e o array contendo a amostra, então deve retornar o índice que corresponde à linha da matriz que seja mais similar ao array amostra (menor valor). A seguir, o programa deve imprimir a classe associada a esse índice mínimo, por exemplo:

```
int indice_min = classificar_1NN(caracteristicas, amostra);
printf('A classe é %s', classes[indice_min]);
```

A similaridade entre arrays pode ser calculada usando uma métrica de distância (ex. distância Euclidiana, Manhattan, etc.). Quanto menor a distância entre dois objetos for, mas similares eles são.

Sejam dois objetos $X_1=(x_{11}, x_{12},...,x_{1n})$ e $X_2 =(x_{21},x_{22},...,x_{2n})$, a distância euclidiana entre X_1 e X_2 é dada por:

$$\text{dist}(X_1, X_2) = \sqrt{\sum_{1}^{n}\left(x_{1i} - x_{2i}\right)^2}$$

Finalmente a **opção [3]** deve encerrar o programa

Desafio (+1 ponto na prova 2)

Crie uma função KNN que receba a matriz de características, um array da amostra e um inteiro positivo k, a função deve encontrar os k elementos mais similares (menores distâncias) e entre eles selecionar a classe maioritária.

Observações:

- O projeto deve funcionar e estar livre de erros de compilação/execução.
- Escreva um programa que seja reutilizável e crie funções genéricas de modo que possa ser adaptado facilmente a outro banco de dados com características similares

Itens do relatório final:

- Introdução
- Distribuição de tarefas/responsabilidades por aluno
- Atividades desenvolvidas:
    - O que foi feito, o que ficou pendente
        - Adicionar os prints do projeto funcionando
    - Dificuldades encontradas
- Conclusões e trabalho futuro