UFAPE

UNIVERSIDADE FEDERAL DO AGRESTE DE PERNAMBUCO

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

EXERCÍCIOS PRÁTICOS DA DISCIPLINA OPTATIVA FUNDAMENTOS DE CIÊNCIA DE DADOS

EXERCÍCIO 1 – INSTÂNCIA 2024.2

PROFESSORES: RYAN AZEVEDO – ASSUERO XIMENES

{ryan.azevedo, assuero.ximenes}@ufape.edu.br

Para cada item da atividade o aluno precisa exibir o código utilizado para obtenção dos resultados. O aluno pode disponibilizar o código *Colab*, *Jupyter* ou *.py*. Observem todos os resultados obtidos.

PARTE 1 – *NumPy*: práticas necessárias com uso da biblioteca *NumPy*

Usando a biblioteca *NumPy* faça: (Para cada tarefa apresente o código *Python* que e o resultado correto da execução).

1) Crie 1 *array* com 3 elementos distintos
2) Crie 1 *array* de zeros
3) Crie 1 *array* de uns
4) Crie 1 *array* de 4 elementos arbitrários
5) Crie 1 *array* a partir de um intervalo de números (sequência)
6) Crie 1 *array* a partir de um intervalo de números e que apresente uma sequência de 3 em 3.
7) Explique o resultado desse código: *np.linspace*(0,22,5)
8) Crie um *array* de 21 elementos e exiba:
    a. Tipo
    b. Número de elementos
    c. Consumo de *bytes* por elementos
    d. Número de elementos
    e. Número de dimensões
9) Criar uma lista A de 3 listas com 3 elementos
10) Copiar a lista do item anterior para um *array* multidimensional chamado multi_B
11) Exibir o número de elementos em cada dimensão do *array* lista A e B

12) Usando o comando *reshape* crie transforme uma lista de 12 elementos em 1 array (3,4)
13) Usando o comando *reshape* transforme o *array* anterior em uma lista de 12 elementos
14) Usando o comando *reshape* transforme a lista anterior em um *array* (2,2,3)
15) Crie um *array* de 21 elementos e apresente:
    a. O elemento 3
    b. O último elemento
    c. os elementos no intervalo de 3 ao 9
16) Crie 1 *array* de 21 elementos usando o comando *arange* e *reshape* em uma mesma linha de comando e faça:
    a. Apresente o elemento 2 (linha 2)
    b. Apresente o elemento 3 da linha 2
    c. Apresente as linhas 0 e 1 e seus elementos
    d. Apresente os elementos da coluna 3 de cada linhas do *array*
    e. Apresente apenas os 3 últimos elementos da 1ª e 2ª linha do *array*
    f. Apresente apenas os 5 últimos elementos de cada linha do *array*
17) Crie 1 *array* de 21 elementos usando o comando *arange* e *reshape* em uma mesma linha de comando, depois de criado altere o primeiro elemento para 51
18) No *array* anterior (3,7) altere apenas as linhas 2 e 3 para que os 3 primeiros elementos de cada seja 0.
19) Crie duas listas e apresente em uma nova lista:
    a. A soma os elementos da lista
    b. A multiplicação dos elementos dessas listas
    c. A divisão dos elementos dessas listas
20) Crie 1 *array* de 24 elementos e apresente:
    a. O maior valor
    b. O menor valor
    c. A soma dos valores
    d. A média dos valores
    e. A soma de cada linha
    f. A soma de cada coluna
21) Crie na forma de polinômio usando o comando poly1d os seguintes polinômios:
    a. $3x + 4$
    b. $4x^3 + 3x^2 + 2x + 1$
    c. $2x^3 + 3$
    d. $x^2 + 2x + 3$
    e. Multiplique os polinômios c e d
    f. Derive o polinômio d
    g. Integre o polinômio d

PARTE 2 – *SciPy*: práticas necessárias com uso da biblioteca *SciPy*

Usando a biblioteca *SciPy* faça: (Para cada tarefa apresente o código *Python* que e o resultado correto da execução).

Usando o submódulo *scipy.misc* faça:

1) Carregue e exiba a imagem "face" em seu formato original.
2) Exiba a imagem anterior em escala cinza
3) Apresente o *array NumPy* referente a imagem.

Operações de Interpolação de Imagens

4) Instale o *scikit-image* no seu ambiente
5) Redimensione a imagem gerada anteriormente para 50% do seu tamanho original
6) Volte a imagem para o tamanho original usando interpolação bilinear (Note que mesmo recuperada a imagem não apresenta a mesma qualidade da imagem original.)

Usando o submódulo *scipy.special* faça:

7) Calcule e apresente o fatorial do número 4 como exemplo
8) Apresente os gráficos das funções de *Besel* e da função de erro *Gaussiana*
9) Apresente o gráfico da função *Gama*
10) Apresente os gráficos dos polinômios de *Legendre*

Usando o submódulo *scipy.stats* (distribuições discretas e contínuas) faça:

11) Apresente os gráficos das funções PDF e CDF (CDF é a integral de PDF (Função de Densidade de Probabilidade) e PDF é a derivada de CDF).
12) Apresente o gráfico de probabilidade da função de massa de probabilidade (PMF) para a distribuição binomial (Probabilidade x Número de Sucessos).
13) Gere uma amostra aleatória e apresente a *estatística t* e o *valor p*
14) Apresente os valores de correlação de *Pearson* e *Spearman* e um gráfico com linha de regressão ajustada sobre os pontos de dados.
    a. A correlação de *Pearson* avalia a relação linear entre duas variáveis contínuas.
    b. A correlação de *Spearman* avalia a relação monotônica entre duas variáveis contínuas ou ordinais.

Parte 3 - Manipulação de Dados Tabulares com *Pandas*

Usando o *pandas* faça:

1) Crie um *Dataframe* (***Cidades***) (Vide Tabela 1)
   - O *Dataframe* deve possuir 5 colunas e 5 linhas. *Obs.* (uma coluna é criada automaticamente e recebe o número referente ao *Id* do registro criado.)

	ESTADO	CIDADE	HABITANTES	MÉDIA SALÁRIAL
0	Paraíba	João Pessoa	604564	1500
1	Rio Grande do Norte	Natal	765298	2000
2	Pernambuco	Recife	890765	3023
3	Ceará	Fortaleza	149087	3200
4	Bahia	Salvador	1920613	2199

Tabela 1: *Dataframe* **cidades**

Obs.:(A primeira coluna foi criada automaticamente)

2) Para o *Dataframe Cidades*, faça/exiba (Usando funções básicas do *Pandas*):
   - O *Dataframe*
   - Apenas as 2 primeiras linhas
   - Apenas as 2 últimas linhas
   - Número de linhas e colunas. *Ex*.:(3, 6)
   - Informações Gerais
   - O nome de todas as colunas
   - As medidas estatísticas para os campos Média Salarial e Habitantes
   - As informações gerais
   - Os tipos de dados
   - Altere o tipo de dados da coluna Habitantes para *float*
   - Insira uma nova coluna chamada Data no formato “dia, mês e ano” e insira 5 datas aleatórias.
   - Renomeei as colunas: Habitantes para **Nº Habitantes** e Data para **Data – Censo**
   - Localize as informações apenas da cidade de Natal
   - Localize as informações das cidades de João Pessoa e Fortaleza
   - Localize as informações entre as cidades de Recife e Salvador
   - Todas as linhas e apenas as duas primeiras colunas
   - As duas primeiras linhas e as 3 primeiras colunas
   - Todas as linhas e apenas as colunas na ordem: Cidade, Média Salarial, Estado e Nº Habitantes
   - Insira uma coluna chamada: **Sigla**, na posição [2] e insira os dados das siglas de cada estado existente
   - Adicione novamente a mesma coluna com nome: Sigla-Repetida e mesmos dados
   - Usando **Pop** exclua a coluna **Sigla-Repetida**

- Adicione novamente a mesma coluna com nome: Sigla-Repetida e mesmos dados
- Usando ***Drop*** exclua a coluna **Sigla-Repetida**
- Os dados que possuem Média Salarial menor que 2100 R$.
- Apenas os dados das cidades do estado de PE.
- As cidades que possuem Número de Habitantes entre 500000 e 1500000
- Ordenar os dados por ordem alfabética da coluna Cidades
- Ordenar os dados por valor do Salário Médio

3) Crie um *Dataframe* (**cidades_salarios**) usando o *Pandas* (Vide Tabela 2)
   - O *Dataframe* deve possuir 3 colunas e 5 linhas. *Obs.* (uma coluna é criada automaticamente e recebe o número referente ao *Id* do registro criado.)

	ESTADO	CIDADE	MÉDIA SALÁRIAL
0	Paraíba	João Pessoa	1500
1	Rio Grande do Norte	Natal	2000
2	Pernambuco	Recife	3023
3	Ceará	Fortaleza	3200
4	Bahia	Salvador	2199

Tabela 2: *Dataframe* **cidades_salarios**

Obs.:(A primeira coluna foi criada automaticamente)

4) Crie um *Dataframe* (**cidades_pop**) usando o *Pandas* (Vide Tabela 3)
   - O *Dataframe* deve possuir 2 colunas e 5 linhas. *Obs.* (uma coluna é criada automaticamente e recebe o número referente ao *Id* do registro criado.)

	CIDADE	HABITANTES
0	João Pessoa	604564
1	Natal	765298
2	Recife	890765
3	Fortaleza	149087
4	Salvador	1920613
5	Teresina	1678943

Tabela 3: *Dataframe* **cidades_pop**

Obs.:(A primeira coluna foi criada automaticamente)

5) Para os *Dataframes* **cidades_salarios** e **cidades_pop**, faça/exiba usando ***merge***:
   - Faça a união (*Inner Join*) dos dois *Dataframes* criados pela coluna **Cidade**
   - Faça a união (*full Outer Join*) dos dois *Dataframes* criados pela coluna **Cidade (Obs:. Registros do tipo NaN serão criados)**
6) Crie dois *Dataframes* (**cidades_df1**) e (**cidades_df2**) respectivamente, usando o *Pandas* (Vide Tabela 4 e 5)
   - O *Dataframe* deve possuir 3 colunas e 5 linhas. *Obs.* (uma coluna é criada automaticamente e recebe o número referente ao *Id* do registro criado.)

	CIDADE	HABITANTES
0	João Pessoa	604564
1	Natal	765298
2	Recife	890765

Tabela 4: *Dataframe* **cidades_df1**

	CIDADE	HABITANTES
4	Salvador	1920613
5	Teresina	1678943

Tabela 5: *Dataframe* **cidades_df2**

- Para os *Dataframes* **cidades_df1** e **cidades_df2**, faça/exiba usando ***concat***:
    - Combine os *Dataframes* em um único não ignorando os *Index*
    - Combine os *Dataframes* em um único ignorando os *Index*

7) Salve o *Dataframe* **cidades** nos seguintes formatos:
    - Cid_01.csv
    - Cid_01.html
    - Cid_01.json

Apresente os códigos necessários e com exemplos para:

8) Ler uma base *csv* de um link e apresentar os primeiros dados dessa base (sugestão: http://files.grouplens.org/datasets/movielens/ml-100k/u.user).
9) Ler uma base *csv* do arquivo do seu computador local e apresentar os primeiros dados dessa base.

Parte 4 – Redes Neurais

1) **MLP:** Usar (Carregar e Importar) a base de dados ***parkinsons*** **(**https://www.kaggle.com/datasets/vikasukani/parkinsons-disease-data-set), e:
   - Normalizar todas as colunas (todos os valores das colunas entre 0 e 1)
   - Separar o *dataset* em X (matriz de *features*) e y (coluna *target*)
   - Gerar as bases de treinamento e teste (Informar porcentagem utilizada)
   - Os alunos devem importar o modelo MLP do sklearn (Usar o MLPClassifier)
   - Escolher e apresentar:
     - Topologia de rede
     - Função de Ativação
     - Número de Execuções (Para uma taxa de acertos estável)
   - Treinar o modelo com os dados de treinamento (Informar porcentagem utilizada)
   - Fazer o predict com os dados de teste
   - Imprimir o percentual de acerto da base de teste

   OBS: **Para a Questão 1 – MLP, apresente os resultados obtidos em gráficos. Apresente pelo menos três gráficos diferentes para os "clientes" visualizarem os resultados. Apresente a topologia da Rede. Crie um relatório apresentando os gráficos e a topologia da RNA (em imagem) utilizada.**

2) **MLP:** Usar (Carregar e Importar) a base de dados ***penguin*** **(API *seaborn*)** – **(**https://www.kaggle.com/datasets/larsen0966/penguins), e:
   - A base deve ser tratada: existem valores nulos e dados categóricos
   - Normalizar todas as colunas (Valores das colunas entre 0 e 1)
   - Separar o *dataset* em X (matriz de *features*) e y (coluna *target*)
   - Gerar as bases de treinamento e teste (Informar porcentagem utilizada)
   - Os alunos devem importar o modelo MLP do *sklearn*
   - Escolher e apresentar:
     - Topologia de rede
     - Função de Ativação
     - Número de Execuções (Para uma taxa de acertos estável)
   - Treinar o modelo com os dados de treinamento (Informar porcentagem utilizada)
   - Fazer o *predict* com os dados de teste
   - Imprimir o percentual de acerto da base de teste

   OBS: **Para a Questão 2 – MLP, apresente os resultados obtidos em gráficos. Apresente pelo menos três gráficos diferentes para os "clientes" visualizarem os resultados. Apresente a topologia da Rede. Crie um relatório apresentando os gráficos e a topologia da RNA (em imagem) utilizada.**