Bootcamp IGTI

Desafio

Módulo 4	Python Avançado

Objetivos

Exercitar os seguintes conceitos trabalhados no Módulo:

- ✔ Scikit-Learning.
- ✔ Programação concorrente.
- ✔ Programação reativa.
- ✔ Pygame

Enunciado

Neste desafio são utilizados todos os módulos apresentados durante o módulo 4 deste bootcamp de desenvolvedor Python. Módulos como o scikit-learn, pandas, threading, rx e pygame são empregados para construir aplicações que utilizem conceitos mais avançados da linguagem python. Desse modo, é possível perceber a vasta aplicabilidade desta linguagem para resolver diversos problemas de diferentes complexidades através da computação.

Atividades

Os alunos deverão desempenhar as seguintes atividades:

1. Acessar a IDE de desenvolvimento desejada. (recomendável, para as questões de 1 a 12 utilizar o próprio google collaboratory)
2. Baixar o dataset presente no link:

https://drive.google.com/drive/folders/1twf6tSeqLqHWviy0vY-R4vwx-NMBZBa3?usp=sharing

3. Responder às questões presentes neste desafio.

Obs:

O dataset utilizado possui as seguintes colunas:

- Sex - gênero do paciente ->Homem = 1, Mulher =0
- Age - Idade do paciente
- Diabetes - Possui diabetes? 0 = Não, 1 = Sim
- Anaemia - Possui anemia? 0 = Não, 1 = Sim
- High_blood_pressure - Possui pressão alta? 0 = Não, 1 = Sim
- Smoking - É fumante? 0 = Não, 1 = Sim
- DEATH_EVENT - evento de morte? 0 = Não, 1 = Sim

❖ Para as perguntas referentes aos modelos utilize:

Algoritmo KNN:

```
clf_KNN = KNeighborsClassifier(n_neighbors=5)
```

Algoritmo Árvore de Decisão

```
clf_arvore = DecisionTreeClassifier(random_state=1)
```

Algoritmo Rede MLP

```
clf_mlp = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 10), random_state=1)
```

- Para a aplicação dos algoritmos, utilize como entrada as colunas: ***Sex, Age, Diabetes, Anaemia, High_blood_pressure, creatinine_phosphokinase, Smoking, ejection_fraction, platelets, serum_creatinine e serum_sodium*** . A saída para os algoritmos deve ser a coluna ***DEATH_EVENT***.
- Utilize, para normalização dos dados, as definições:

```
normaliza = MinMaxScaler() #objeto para a normalização
entradas_normalizadas=normaliza.fit_transform(entradas)
```

- Utilize, para divisão entre treinamento e teste do algoritmo, as definições:

```
train_test_split(entradas_normalizadas, saida,
test_size=0.30,random_state=42)
```

- Utilize esta sequência de operações para chegar no resultado final: divida os dados entre entrada e saída, normalize apenas as entradas utilizando o ***MinMaxScaler*** e, depois, aplique a divisão entre treinamento e teste com o ***train_test_split***.
- Utilize os dados de "teste" para avaliar as previsões de classificação dos modelos.

Para as questões de concorrência, utilize a função abaixo como a "tarefa" a ser realizada pelas threads.

```
def contador():
      x = 1000000000
      while x > 0:
            x -= 1
```

Para as chamadas sequenciais utilize o protótipo:

```
def imple_sequencial():
  contador()
  contador()
```

Para as threads utilize o protótipo:

```
def imple_concorrente():
  thread_1 = threading.Thread(target=contador)
  thread_2 = threading.Thread(target=contador)
```

Para a implementação com o tempo (dormir) das threads, utilize os códigos abaixo.

```
import time
import random

time.sleep(random.randint(1,20))
```

Para as questões de programação reativa, utilize o **observable** recebendo o streaming de dados e a inscrição para o **observer** como abaixo:

- `source = rx.from_iterable([5,4,3,2,1]) #streaming`
- `disposable=source.pipe(`

`.subscribe(`

```
    on_next=lambda i: print("on_next: {}".format(i)),

    on_completed=lambda: print("on_completed"),

    on_error=lambda e:print("on_error: {}".format(e))

) #inscrição do observer
```

Dica: Utilize, como base, a implementação presente na primeira aula sobre programação reativa.

Para as questões referentes ao Pygame, utilize o esboço de código abaixo:

```python
# coding: iso-8859-1 -*-
import pygame
from pygame.locals import *
from sys import exit

pygame.init()

screen = pygame.display.set_mode((720, 640))
pygame.display.set_caption("Desafio-Módulo 4")


while True:

    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            exit()

        screen.fill((255,0,255))

        x, y = pygame.mouse.get_pos()

        print(x,y)

    pygame.display.update()
```

Respostas Finais

Os alunos deverão desenvolver a prática e, depois, responder às seguintes questões objetivas:

1. Sobre as quantidades de instâncias e características presentes no dataset é CORRETO afirmar

x	Existem 299 instâncias e 13 características
	Existem 13 instâncias e 299 características
	Existem 189 instâncias e 10 características
	Existem 10 instâncias e 189 características

2. Quantos tipos de dados diferentes existem no dataset? Considerando apenas a carga utilizando o módulo pandas.	
x	2 tipos diferentes de dados
	3 tipos diferentes de dados
	4 tipos diferentes de dados
	1 tipo diferente de dado

3. Qual era a idade (age) média dos pacientes que faleceram (DEATH_EVENT=1)?	
x	65,21 anos
	60,83 anos
	58,27 anos
	73,87 anos

4. Dentre os pacientes que sobreviveram (DEATH_EVENT=1), quantos são do sexo feminino (sex=0)?	
x	71
	114
	87
	65

5. Após dividir as colunas do *dataframe* entre entrada e saída, aplicar a normalização dos dados como apresentado no enunciado (MinMaxScaler()) e dividir esses dados entre treinamento e teste, aplique o algoritmo **KNN**. Qual é, aproximadamente, a acurácia do modelo?	
x	0,61
	0,76
	0,48

	0,81

6. Após dividir as colunas do *dataframe* entre entrada e saída, aplicar a normalização dos dados como apresentado no enunciado (MinMaxScaler()) e dividir esses dados entre treinamento e teste. Aplique os algoritmos **KNN**, **Árvore de Decisão** e **MLP**. Qual dos algoritmos apresentou maior acurácia?	
x	MLP
	KNN
	Árvore de Decisão
	Nenhum dos modelos obteve resultado superior a 50% de acurácia

7. Utilizando a função mostrada no enunciado, implemente duas chamadas sequenciais e outra chamada sendo realizada por duas threads de maneira concorrente. Sobre essas duas diferentes formas de implementação é CORRETO afirmar	
x	O tempo de execução através da concorrência é inferior à sequencial, pois cada thread realiza as operações através do chaveamento de contexto.
	O tempo de execução através da concorrência é inferior à sequencial, pois cada thread realiza as operações através do processamento paralelo.
	O tempo de execução através da concorrência é superior à sequencial, pois cada thread realiza as operações através do chaveamento de contexto e compartilham recursos.
	O tempo de execução através da concorrência é superior à sequencial, pois cada thread realiza as operações através do paralelismo, o que sobrecarrega o processador.

8. Utilizando a função mostrada no enunciado, implemente as duas chamadas sequenciais e outra chamada sendo realizada por duas threads de maneira concorrente. Adicione à chamada concorrente um tempo randômico para cada thread, como apresentado no enunciado. Após a construção dessas implementações é CORRETO afirmar

x	Após a adição desse tempo aleatório sem processamento (dormir), não é possível garantir que a implementação realizada com as duas threads seja realizada em um tempo menor.
	Independente do tempo escolhido para deixar as threads sem realizar um processamento (dormir), a execução sequencial sempre será mais lenta.
	O processamento paralelo não poderia ser utilizado após a adoção desse tempo sem processamento (dormir), pois no paralelismo não é possível ficar sem realizar um processamento.
	Quando adicionamos um período sem processamento (dormir), o sistema operacional não altera o estado da thread, assim, o tempo de processamento não é modificado.

9. Utilizando os códigos apresentados no enunciado deste desafio, referente às questões de programação reativa, qual das afirmativas abaixo representa uma possibilidade de transformação sobre os dados enviados pelo **Observable** para que o **Observer** receba apenas números pares?	
x	`ops.filter(lambda i:i%2==0)`
	`ops.map(lambda i:i%2==0)`
	`ops.map(lambda i:i/2==0)`
	`ops.filter(lambda i:i/2==0)`

10. Utilizando os códigos presentes no enunciado deste desafio, referente às questões de programação reativa, se adicionarmos apenas as operações de map e filter abaixo, qual será o resultado?
`ops.map(lambda i: i if i<3 else 0),`
`ops.filter(lambda i: i>0),`	

x	Os valores 2 e 1 seriam exibidos no “on_next”
	Nada seria exibido no “on_next”
	Apenas os números pares seriam exibidos no “on_next”
	Seria exibido uma mensagem no “on_error”

11. Utilizando os códigos presentes no enunciado deste desafio, referente às questões de programação reativa, substitua a lista [5,4,3,2,1] pela lista [5,4,3,"2",1]. Adicione apenas a operação de filtro `ops.filter(lambda i: i%2==0),`. Qual será a saída apresentada após essas mudanças?	
x	Será apresentado no "on_next" o segundo número da lista e um "on_erro".
	Será apresentado apenas no "on_next" os valores pares (4 e 2).
	Será apresentado apenas no "on_next" os valores primos presentes na lista.
	Será apresentado apenas no "on_error" os valores pares menores que 2.

12. Sobre o código presente no enunciado, referente às questões do Pygame, é INCORRETO afirmar	
x	Mesmo que o cursor do mouse seja posicionado fora da tela do jogo, será exibido um "print" com a posição atual.
	Para alterar as cores do plano de fundo da tela é necessário modificar os parâmetros do método "fill".
	No terminal/console será exibida as posições que o cursor do mouse está na tela do jogo.
	Se retirarmos a linha 25 não ocorre um erro de execução, entretanto, a tela não será atualizada.

13. Utilize como base o esboço de código para as questões do Pygame. Quais modificações seriam necessárias para capturar, de maneira contínua, os eventos de teclas digitadas?	
x	Adicionar, entre as linhas 18 e 20, a captura dos eventos através do event.type==KEYDOWN.
	Retirar a captura e o tratamento dos eventos presentes entre as linhas 15 e 17.
	Adicionar, entre as linhas 10 e 11, a captura dos eventos através do event.type==KEYDOWN.

	Retirar o código presente entre as linhas 14 e 15 e adicionar a captura de eventos através do event.type==KEYDOWN

14. Utilizando o código referente às questões do Pygame, adicione, à tela, um retângulo vermelho de dimensões 10x10 pixels. Quais alterações devem ser realizadas?	
x	Adicionar, entre as linhas 12 e 14, a chamada pygame.draw.rect(screen, (255,0,0), [360,320,10,10]).
	Adicionar, entre as linhas 9 e 11, a chamada pygame.draw.rect(screen, (255,0,0), [360,320,10,10]).
	Adicionar, entre as linhas 12 e 14, a chamada pygame.draw.rect(screen, (0,0,0), [720,640,10,10]).
	Adicionar, entre as linhas 9 e 11, a chamada pygame.draw.rect(screen, (255,0,0), [0,0,10,10]).

15. Utilizando o código referente às questões do Pygame e o retângulo criado ao centro da tela, quais alterações poderiam ser realizadas para realizar a movimentação deste retângulo através das teclas direcionais do teclado?	
x	Adicionar, entre as linhas 17 e 19, a captura do evento de pressionar uma tecla, identificar as teclas pressionadas e atualizar a posição do retângulo.
	Retirar a linha 14, adicionar, entre as linhas 14 e 17, a captura do evento de pressionar uma tecla, identificar as teclas pressionadas e atualizar a posição do retângulo.
	Adicionar, entre as linhas 17 e 19, a captura do evento de pressionar uma tecla, identificar as teclas pressionadas e atualizar a posição do retângulo e retirar a linha 25.
	Retirar a linha 25, adicionar, entre as linhas 14 e 17, a captura do evento de pressionar uma tecla, identificar as teclas pressionadas e atualizar a posição do retângulo.