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. 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 se 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, reatinine_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:

```
# 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: