Processo Seletivo Grupo Turing 2020

Case Técnico - Nível Intermediário

Olá candidatx! Seja muito bem vindo ao processo seletivo Grupo Turing 2020! Nessa segunda fase do nosso processo seletivo, você terá que resolver um case técnico. São várias perguntas, incluindo uma dissertação e 1 desafio, sendo que você pode escolher quais e quantas questões irá responder. Algumas questões são propositalmente desafiadoras, então saiba o que atacar, padawan. O intuito aqui é entender o seu raciocínio e qualidade de código, bem como o seu desenvolvimento no processo. Portanto, **não se preocupe ou se desespere em fazer tudo**. Nós queremos ver até onde você chega, e já adiantamos, será desafiador!

Lembrete: isso não é uma prova da USP, sinta-se livre para procurar guias e vídeos na internet para te ajudar.

Nesse mesmo espírito, **não se preocupe se você está um pouco enferrujado em programação**. Vamos verificar isso durante a dinâmica de grupo, que ocorrerá antes da correção do case, para que possamos avaliá-lo adequadamente.

Caso tenha alguma dúvida, é só perguntar para a gente no nosso e-mail turing.usp@gmail.com, na nossa página https://www.facebook.com/grupoturing.poliusp/ ou mandar uma mensagem para algum dos membros da equipe abaixo:

Felipe Azank	+55 (19) 99662-3900
Leonardo Murakami	+55 (11) 97683-8455
Ariel Guerreiro	+55 (11) 94331-1181
Fernando Matsumoto	+55 (19) 99850-2800

Tentaremos responder o quanto antes! Além disso, **teremos monitorias nas próximas semanas,** enviaremos hora e local por e-mail. O envio da maioria dos arquivos será feito em **Python 3**. No entanto, nas questões de lógica de programação (2 primeiros exercícios), aceitaremos também o uso de C, C++, R, Ruby.

Para algumas das questões, será utilizado o jupyter notebook, como será especificado na seção *formato da submissão*. O método mais prático para instalar jupyter no seu computador é através do Anaconda, uma plataforma que reúne vários softwares de programação em python. Para instalá-lo basta seguir o tutorial no link abaixo:
https://www.datacamp.com/community/tutorials/installing-anaconda-windows

Você também pode utilizar o jupyter notebook no Google Collab:
https://colab.research.google.com/notebooks/welcome.ipynb

Também é importante lembrar que quanto mais bem comentado o código melhor, pois facilitará o entendimento do seu código e da lógica por trás do mesmo.

Para a parte de programação, recomendamos o uso das bibliotecas: pandas, numpy, scikit learn, matplotlib e seaborn. Você pode seguir alguns de nossos tutoriais (abaixo) e outros tutoriais listados também dentro de cada questão:

- Tutorial de python:
  - Parte 1: https://medium.com/turing-talks/turing-talks-4-python-parte-1-29b8d9efd0a5
  - Parte 2: https://medium.com/turing-talks/turing-talks-5-python-parte-2-97198bae699e
- Videoaulas de python:
  https://www.youtube.com/playlist?list=PL-tx-k-UlaL6KaKRR1NWbTxZWrE5Lc1Eu
- Jupyter notebooks e bibliotecas (pandas, numpy, matplotlib):
  https://medium.com/turing-talks/turing-talks-6-data-science-libraries-6c2599838b3e

Formato de Submissão

Quando terminar seu case, envie-o para o nosso e-mail (turing.usp@gmail.com) com o assunto **"Case Intermediário PS 20 - *Seu nome completo*"** e, em anexo, um arquivo zip com o seu nome e a(s) sua(s) resposta(s). **Não aceitaremos múltiplas submissões.**

Você deve seguir o seguinte formato:

- As respostas dos exercícios 1 e 2 (lógica de programação) podem ser entregues em Python (.py ou ipynb), C (.c), C++ (.cpp), R (.r) e Ruby (.rb ou .ruby).
- As respostas dos exercícios 3 e 4 (análise de dados e algoritmos de IA) devem ser entregues em Jupyter Notebook (.ipynb). Para um exemplo, veja o notebook analise_wine.ipynb presente em nosso GitHub.
- A resposta do exercício 5 (dissertação) deve ser entregue em pdf.

Os nomes dos arquivos e pastas devem identificar as questões que eles resolvem, **conforme a estrutura abaixo** (só inclua as partes que você resolver, não precisam estar completas para manda-las):

```
andre_barbieri.zip
├ 1_logica1
|  ├ item_a.py (ou demais formas)
|  ├ item_b.py (ou demais formas)
|  └ item_c.py (ou demais formas)
├ 2_logica2
|  ├ item_a.py (ou demais formas)
|  ├ item_b.py (ou demais formas)
|  └ item_c.py (ou demais formas)
├ 3_analise
|  └ analise.ipynb
├ 4_desafio_ia
|  └ ia.pdf
└ 5_dissertacao
   └ dissertacao.pdf
```

Observações:

- Se quiser, sinta-se livre para dividir os seus programas em mais de um arquivo. Nesse caso, os arquivos principais devem seguir a estrutura ao lado; a nomenclatura dos outros arquivos fica a seu critério.
- Recomendamos dar uma passada em todas as partes antes de começar a fazer o case para escolher o que você se sente mais confortável em fazer! E não fique assustado com o tamanho dos enunciados, a maioria do conteúdo foi colocado justamente para ajudar na realização da tarefa.

Parte 1: Lógica de programação - Exercícios 1 Verificação e Criação de CPF

`Formato: .py, .ipynb, .c, .cpp, .r, .rb ou .ruby`

Nessa etapa, você utilizará seus conhecimentos básicos de lógica de programação para resolver um problema relacionado à validação de CPFs.

O Cadastro de Pessoas Físicas (CPF) é um documento da Receita Federal do Brasil criado em 1968 que identifica unicamente pessoas naturais. Cada pessoa inscrita no registro é identificada por um número único de 11 dígitos decimais.

O formato do número de identificação consiste no agrupamento dos 9 primeiros dígitos, normalmente separados de 3 em 3 por um ponto (.), seguidos por um hífen (-) e os 2 últimos dígitos, chamados *dígitos verificadores*. Por exemplo: **123.456.789-10**, sendo também possíveis as formas somente com o hífen (**123456789-10**) ou somente com os dígitos (**12345678910**).

ITEM A

a. Por meio do número de um CPF é possível identificar de qual estado o documento foi emitido. Para tal, basta verificar o 9º dígito do documento (antes dos dígitos verificadores). Cada estado (ou conjunto de estados) possui seu número correspondente. Por exemplo, o CPF **123.456.789-10** é do estado do Paraná ou de Santa Catarina.

Código de Área	Estados representantes
0	Rio Grande do Sul
1	Distrito Federal, Goiás, Mato Grosso do Sul e Tocantins
2	Pará, Amazonas, Acre, Amapá, Rondônia e Roraima
3	Ceará, Maranhão e Piauí
4	Pernambuco, Rio Grande do Norte ,Paraíba e Alagoas

5	Bahia e Sergipe
6	Minas Gerais
7	Rio de Janeiro e Espírito Santo
8	São Paulo
9	Paraná e Santa Catarina

Neste exercício, deve fazer um programa que receba uma lista de CPFs e devolva uma lista com somente os CPFs que foram emitidos pelo estado de **Minas Gerais**.

Importante: Os CPFs recebidos pelo seu programa estarão na forma sem pontos ou hífen e também devolverá os CPFs desta forma.

Exemplos:

```
Input: [17649147673,
        34422488090,
        79537935943,
        89883888368,
        77812698610,
        53266333292]


Output: [17649147673,
         77812698610]
```

```
Input: [27995647808,
        40427300193,
        81357249268,
        06364276687,
        90189191520]



Output: [06364276687]
```

ITEM B

b. Os dois últimos dígitos do CPF (após o hífen) são chamados de dígitos verificadores pois são capazes de verificar se o CPF fornecido é real ou falso. Os números são determinados a partir dos outros, de acordo com a seguinte forma:

1º Dígito Verificador (10º número): Para definir este número somamos os **9** algarismos anteriores, usando como pesos a sequência 10, 9, 8, … , 3, 2. Em seguida, dividimos a

soma por 11 e pegamos seu **resto**. Se o resto for 0 ou 1, o dígito será **0**. Caso contrário, o dígito será o resultado de **11 - resto**.

Por exemplo, com o número **196251015**:

(**1** x 10 + **9** x 9 + **6** x 8 + **2** x 7 + **5** x 6 + **1** x 5 + **0** x 4 + **1** x 3 + **5** x 2) mod 11 = 3

Portanto o 1º DV é 11 - 3 = **8**.

2º Dígito Verificador (11º número): Para definir este número, somamos os **10** algarismos anteriores, usando como pesos a sequência 11, 10, 9, … , 3, 2. Em seguida, dividimos a soma por 11 e pegamos seu **resto**. Se o resto for 0 ou 1, o dígito será **0**. Caso contrário, o dígito será o resultado de **11 - resto**, assim como foi feito para o dígito anterior.

Mantendo o mesmo exemplo, agora com o 1ºDV: **1962510158**:

(**1** x 11 + **9** x 10 + **6** x 9 + **2** x 8 + **5** x 7 + **1** x 6 + **0** x 5 + **1** x 4 + **5** x 3 + **8** x 2) mod 11 = 5

Portanto, o 2º DV será 11 - 5 = **6**, com o CPF completo sendo **19625101586**.

Neste exercício, deve ser feito um programa que recebe um CPF e responde se este se trata de um CPF verdadeiro ou não, utilizando-se do método para a obtenção dos dígitos verificadores.

OBS: Esse código deve receber os CPFs tanto na forma de inteiro (int) quando na forma de string (str).

Exemplos:

```
Input: '63662854546'
Output: True

Input: 13360069510
Output: False
```

```
Input: 133600695
Output: False

Input: '34821223503'
Output: True
```

ITEM C

c. Neste exercício, deve ser criado um programa que gera CPFs aleatórios **válidos**, seguindo as regras para os CPFs estabelecidas na letra b.

   OBS: você pode utilizar o seu código que foi desenvolvido no item b

Parte 1: Lógica de programação -

Exercícios 2: o enigma de Flavius Josephus

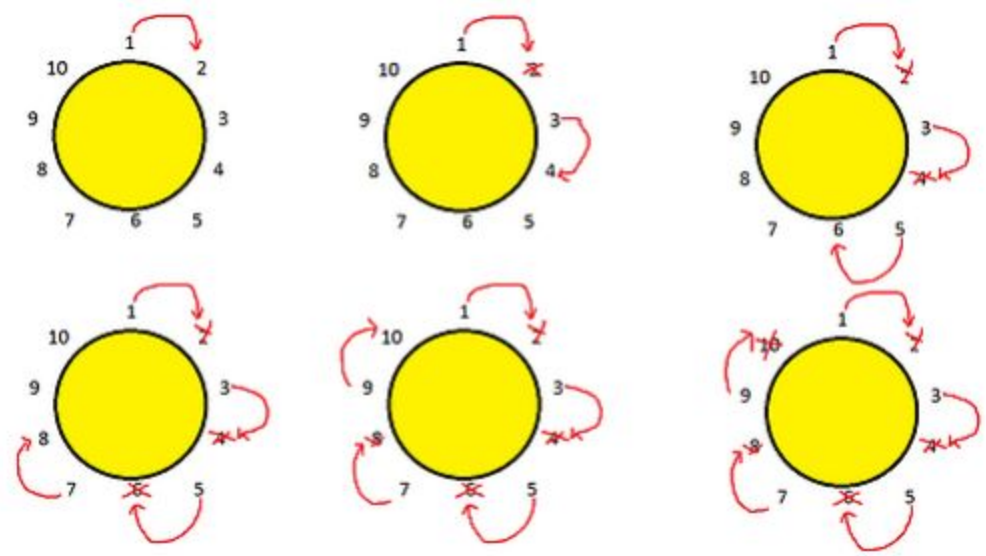
`Formato: .py, .ipynb, .c, .cpp, .r, .rb ou .ruby`

O enigma de Flavius Josephus decorre da seguinte história:

“Um dia, um grupo de soldados judeus, que estavam sendo perseguidos por soldados romanos, acabaram sendo encurralados em uma caverna. Sem saber o que fazer, e sem o desejo de viverem como escravos, os judeus optaram pelo suicídio coletivo do grupo. Entretanto, como o ato de tirar a própria vida não era permitido nos preceitos do Judaísmo, eles desenvolveram um método para que uma pessoa seja morta por vez, até que se sobrasse um.

A lógica consiste na seguinte maneira, as N pessoas presentes são dispostas em círculo e são numeradas de 1 até N.

A primeira pessoa (1) executa a pessoa que está ao seu lado (2), em seguida, a próxima pessoa viva (3) executa o soldado seguinte (4) e assim por diante, até sobrar uma única pessoa (vide o exemplo com N=10):



Problema: Josephus, que estava entre os soldados judeus, não queria ser morto, portanto, ao descobrir a quantidade N de pessoas, ele precisa escolher a posição que garanta a sua sobrevivência (sendo a pessoa "que sobra").

ITEM A

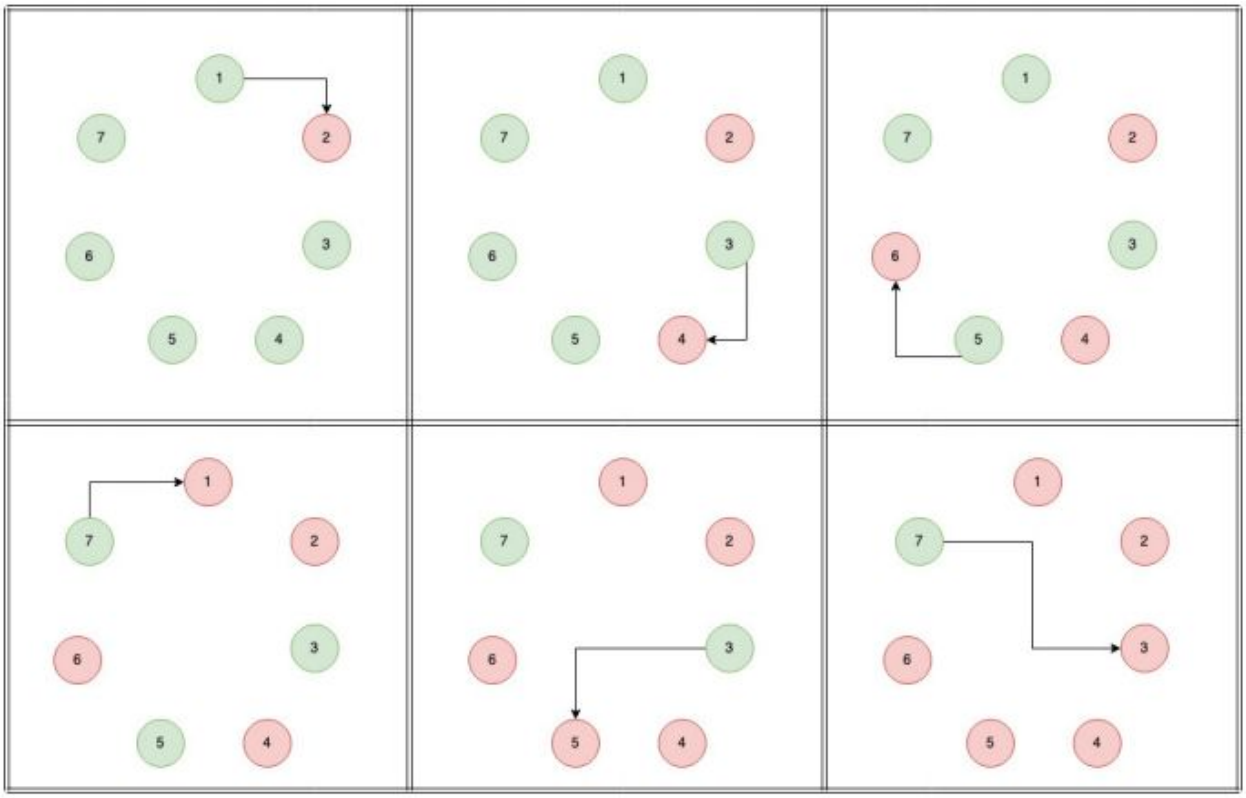
Crie uma função que, dado um array de tamanho variável "n", e uma posição "k" (k < n), transforma o elemento "k" do array no index inicial (vide exemplo em pseudo-código).

```
Ex: (para todos os exemplos considere index inicial = 0)

array circulo = [1, 3, 4, 5, 7, 10]
int k = 3
reset_array(circulo, k) --> returns [5, 7, 10, 1, 3, 4]
```

ITEM B

Crie uma função que, dado um número "n" de pessoas no círculo, retorna a posição que sobreviveria ao problema de Flavius Josephus (vide exemplo em pseudo-código).

```
Ex:

int n = 7
solve_josephus(n) --> returns 6 (pessoa 7, index 6)
```

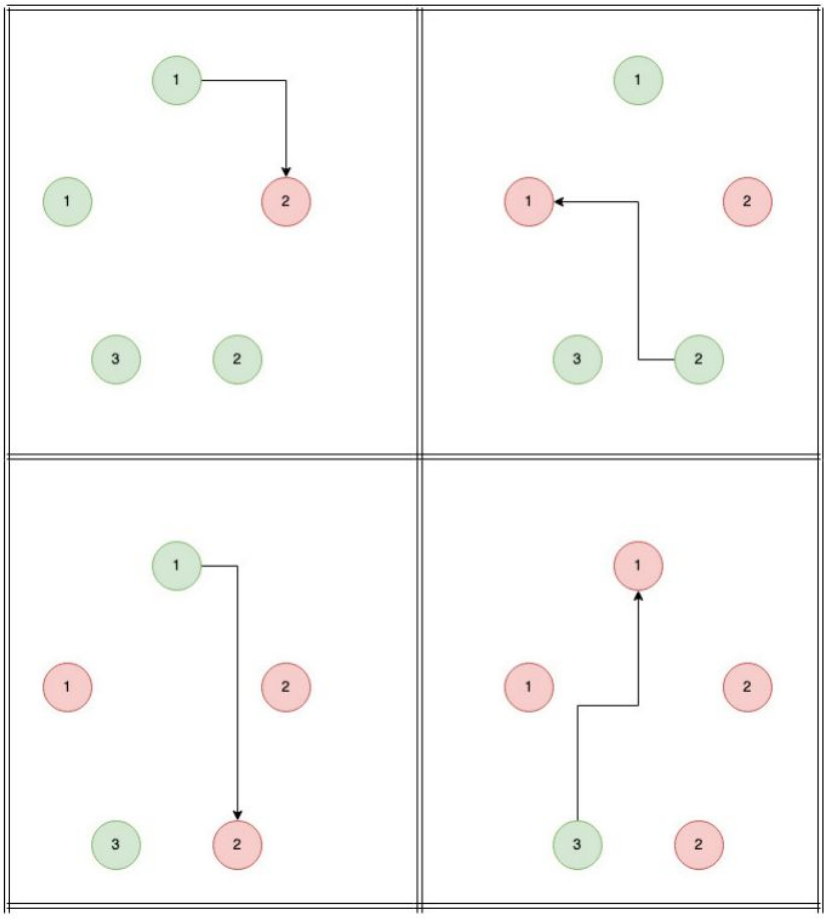


ITEM C

Existe uma outra maneira possível que segue a lógica do problema original, porém, ao invés de sempre matar a pessoa ao seu lado, cada integrante recebe um número "n" e, quando tiver que eliminar uma pessoa, elimina aquela a "n" passos de si, não contando a si mesmo na distância (caso o número passe o tamanho de pessoas). Assim, a pessoa que recebeu o número 1 executa a pessoa ao seu lado, a que recebeu o número 2, executa a pessoa a duas posições de distância, e assim por diante.

OBS: o número recebido pela pessoa pode ser repetido entre as outras pessoas do grupo, como se pode ver no exemplo abaixo.

```
Ex:

array circulo = [1,2,2,3,1]
solve_var_josephus(circulo) --> returns 3 (index 3)
```



Parte 2: Análise de Dados

`Formato: .ipynb`

Nessa etapa, você deverá realizar uma análise de dados básica. Para isso, você pode consultar, além de outros recursos disponíveis na internet:

- O exemplo analise_wine.ipynb. Esse exemplo mostra uma análise básica de um dataset de vinhos tintos. O objetivo é identificar como as características do vinho afetam a sua qualidade.
- O nosso post sobre visualização de dados no medium: https://medium.com/turing-talks/turing-talks-9-visualiza%C3%A7%C3%A3o-de-dados-93df670d479

O Dataset:

O quanto uma pessoa ganha depende de diversos fatores: nível escolar, idade, gênero, ocupação e etc. Tendo isso em vista, nesse exercício, vamo analisar um censo realizado nos EUA que analisou diversos fatores e anotou se essa pessoa ganha mais ou menos de 50 mil dólares por ano.

Seu objetivo, como Data Scientist, é fazer uma análise das colunas e informar insights que você encontrar nessa jornada, se baseie em nosso post do Medium e no notebook de exemplo, mas você pode utilizar outras funções e métodos para extrair mais informações úteis.

OBS: Não se preocupe em tirar insights “milagrosos”, busque utilizar e procurar métodos interessantes e **comunique** suas descobertas em forma de comentários. Estamos testando mais o seu esforço do que seus resultados aparentes.

Link do dataset: https://www.kaggle.com/wenruliu/adult-income-dataset

Divida seu notebook em duas partes:

Análise de Dados Numérica

Nessa primeira parte da análise de dados, você deve analisar somente as features numéricas (principalmente age, educational-num e hours-per-week). O seu objetivo é, de forma similar ao que foi feito no exemplo analise_wine.ipynb, analisar cada feature individualmente, assim como as relações entre as features. No entanto, a sua análise deve ir um pouco além do que foi mostrado no exemplo (pense, por exemplo, em utilizar outros tipos de visualizações e identificar outliers). Para isso, recomendamos que você veja o nosso post de visualização de dados (acima).

Análise de Dados Categórica

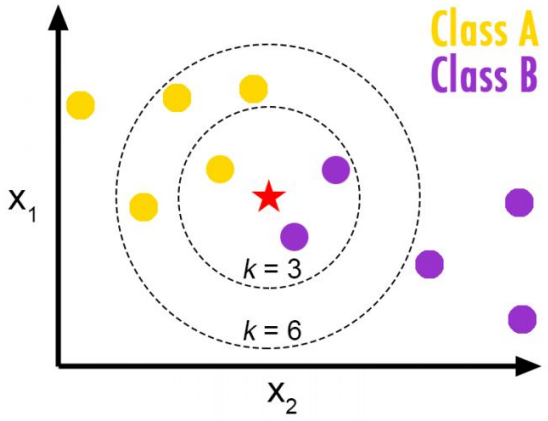
Agora que você já analisou as features numéricas, é hora de analisar as features categóricas, como marital-status, education, workclass e race. Diferentemente das features numéricas, os valores dessas features são restritos a um conjunto de categorias. Você novamente deve analisar as features categóricas individualmente, assim como as relações entre elas. Além disso, nessa parte você também deve analisar as relações de features numéricas com categóricas.

Desafio: Algoritmo de IA

`Formato: .ipynb`

Nessa etapa, o seu desafio vai ser implementar um algoritmo de IA chamado KNN para a classificação do dataset “Adult Income Dataset”, cujo objetivo é avaliar se uma pessoa ganhará mais ou menos de 50 mil dólares em um ano, com base em dados como a idade, escolaridade, estado civil, entre outros.

O k-nearest neighbors (KNN) é um poderoso método de classificação e predição de dados, sendo muito popular e de fácil implementação. Este algoritmo compara os novos elementos com os dados fornecidos na base de treino, que estão salvos na memória. O objetivo do KNN é categorizar esses dados baseado na sua proximidade e similaridade com os dados de treino. O coeficiente K define quantos vizinhos o algoritmo deve analisar. Para predizer a classificação de algum ponto, algoritmo então encontra os K pontos da base de treino mais próximos desse ponto e escolhe a categoria mais comum.



No exemplo acima, para K=3, o exemplo (estrela) seria classificado de classe B, enquanto para K=6, o exemplo seria de classe A. X1 e X2 são características desses elementos (features), que são usadas para posicioná-los dentro do espaço vetorial (no exemplo, 2D).

Considerações importantes:

- Para maior K, pode haver uma melhora na predição dos dados, mas há um maior custo computacional.
- Nem sempre a adição de mais características irá impactar numa melhora de eficiência, uma vez que há o risco de ocorrer overfitting.

Seu objetivo será utilizar o dataset fornecido para criar um modelo de classificação, utilizando o algoritmo de KNN. Você também pode tentar otimizar o modelo, por meio de análise de dados que você fez no exercício anterior, para identificar as features mais úteis, possíveis outliers e etc.

OBS: No dataset, existem dados numéricos e categóricos. A aplicação do algoritmo pode ser feita somente utilizando os dados numéricos, basta retirar as colunas de dados categóricos usando a função do pandas DataFrame.drop. Contudo, caso você queira utilizar os dados

categóricos para a previsão no modelo, sinta-se livre para procurar como fazer isso, ou apenas checar o último guia abaixo.

Tutoriais:

- https://medium.com/turing-talks/turing-talks-n%C3%BAmero-do-post-k-nearest-neighbors-3be880c9b9d1 → Introdução a knn (com scikit-learn) com exemplo
- https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7 → o que é knn
- https://towardsdatascience.com/implementing-k-nearest-neighbors-with-scikit-learn-9e4858e231ea → implementação de knn usando scikit learn
- http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html → documentação KNN scikit learn
- https://towardsdatascience.com/encoding-categorical-features-21a2651a065c → Como transformar dados categóricos em dados numéricos

Módulo recomendado do scikit learn:

```
from sklearn.neighbors import KNeighborsClassifier
```

Dataset:

- Adult Income Dataset → https://www.kaggle.com/wenruliu/adult-income-dataset

  Prever *income* com base nas outras colunas do dataset.

Dica: nos links, as variáveis `x` e `y` geralmente se referem as features (colunas que usaremos para prever algo) e a target (coluna que vamos separar para prever algo de fato). Por exemplo, para o dataset Iris:

```
import pandas as pd
iris = pd.read_csv('iris.csv')
x = iris[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
y = iris['Species']
```

Obs: ainda que as bibliotecas deixem possível utilizar os algoritmos sem entender como eles funcionam, é importante pelo menos ter uma ideia do que eles fazem.

Parte 4: Dissertação

`Formato: .pdf`

Cada vez mais, os algoritmos de inteligência artificial estão presentes no cotidiano das pessoas, seja na sugestão de produtos em comércio eletrônico, na escolha de qual caminho seguir pelo GPS, ou mesmo em aplicações que se tornarão mais comuns, como os carros autônomos. Dada a importância que a IA vem ocupando na sociedade, não basta que os cientistas, engenheiros e desenvolvedores que trabalham com IA sejam competentes em suas habilidades técnicas, mas também, devem ter forte senso crítico, assim como devem ser capazes de perceber o impacto que essas tecnologias causam, além de enxergarem os vieses que podem ser passados na hora de treinar os algoritmos.

Nesta tarefa, você irá redigir um **texto dissertativo** sobre **uma das obras fictícias listadas abaixo** que aborda aplicações de IA. Para isso, escolha uma das obras e, usando sua criatividade e conhecimento de mundo, desenvolva ideias acerca das questões éticas que a obra levanta sobre IA e suas possíveis relações ou implicações para a vida real.

Não nos atentaremos a questões gramaticais e ortográficas, porém lembre-se de que um texto escrito de acordo com a norma culta é capaz de transmitir ideias mais complexas de uma forma mais clara.

Obras para análise: Ex-machina, Her, Blade Runner.

Formatação: fonte Times New Roman ou Arial, tamanho 12, espaçamento 1,5

Número de linhas: entre 10 e 30 linhas

Formato de entrega: PDF com o nome especificado na seção *Formato de Submissão*

Critérios de avaliação:

- Respeitar os limites mínimo e máximo de linhas.
- Atenção ao tema proposto.
- Discussão de questões éticas das aplicações de IA.
- Relação entre a obra e a forma como IA é retratada nela com questões da vida real (por exemplo, notícias).
- Coerência entre o que a obra apresenta e as questões da vida real que o candidato abordar em seu texto.