Um dos principais dados que podem ser extraídos do nosso conjunto de dados, é entender os motivos pelos quais as pessoas escolhem o Windows, que é o sistema operacional do nosso grupo.

Para fazer isso, é necessário tratar os dados já que todas as opções selecionadas pelas pessoas que responderam o questionário vieram em uma só linha, separada por vírgula, utilizando a linguagem Python e a biblioteca pandas. Essa biblioteca serve para tratar dados em forma de tabelas no geral, é construída em cima da biblioteca NumPy, que manipula vetores e matrizes de forma muito eficiente, por ser construída em cima do C, então para casos com um número muito grande de dados, mesmo que não seja o caso aqui, a execução das tarefas é muito rápida, e o Pandas adiciona a isso mais métodos que são úteis no tratamento de dados.

E, para deixar as informações extraídas ainda melhores, permitindo a visualização delas, foi utilizado a biblioteca Matplotlib, também construída em cima do Numpy, que permite a criação de gráficos extremamente customizáveis.

O bloco de código abaixo possui a importação das bibliotecas, a leitura do arquivo excel com todos os dados, e a transformação dos dados em Excel em um Pandas DataFrame com os dados que vamos utilizar agora, que no caso são as características que as pessoas consideram ter influenciado na utilização dos sistemas operacionais, mas apenas com as linhas em que a pessoa utiliza o Windows. Além disso, no caso em que a pessoa não citou nenhuma característica, o programa vai preencher com “None”, para facilitar o tratamento desses valores.

Obs: As linhas ficam diferentes passando para word pelo tamanho disponível, mas vai estar disponível no drive o código completo.

import matplotlib as mpl

from matplotlib import pyplot as plt

import pandas as pd

import numpy as np

#leitura dos dados do arquivo excel com as respostas

dados = pd.read\_excel('questionario.xlsx')

# Selecionando somente as linhas em que a pessoa respondeu que o SO do computador é o Windows,e inserindo "None"

# quando a pessoa não respondeu nada. Além disso, está selecionando somente a coluna das características que

# influenciaram a compra e resetando o index das linhas, já que as que não são index não serão selecionadas

caracteristicas\_influenciaram = dados[dados['SO computador']=='Windows']['caracteristicas SO computador influenciou compra'].reset\_index(drop=True)

caracteristicas\_influenciaram.fillna('None', inplace=True)

No próximo bloco de código, foram realizadas algumas operações para extrair quais eram as respostas únicas da coluna com as características, e a quantidade de vezes que apareceram. Também poderia ter sido usado o Numpy para melhor performance, mas por ser um conjunto de dados pequeno, e só ser usado nessa parte, foi usado a lista do próprio Python mesmo. Também foi passado essas respostas únicas e quantidade de vezes que aparecem em um DataFrame Pandas.

#Transforma a coluna em uma lista, transforma todos os valores em string, junta todas as linhas em um só string separando

# cada linha da lista por ',', aplicando o método split para criar uma só lista em que cada elemento entre vírgulas vira uma linha,

# por fim tira os espaços que podem ter ficado na frente ou atrás das palavras com o strip, rodando um loop for para fazer isso.

# Também transforma em set para fazer a operação no loop for depois

options = (option.strip() for option in ','.join(map(str, caracteristicas\_influenciaram.tolist())).split(','))

count\_dict = {}

# utiliza um loop for para percorrer todas as opções da coluna e somar as vezes que aparecem na coluna e insere em um dicionario

# do python a opcao e a quantidade de vezes que aparece

for option in options:

    count = (caracteristicas\_influenciaram.str.contains(option)).sum()

    count\_dict[option] = count

# transforma o dicionário em um DataFrame Pandas para trabalhar melhor com ele na

# Matplotlib, além de ordenar os valorespelo número de vezes que aparecem

data = pd.DataFrame.from\_dict(count\_dict, orient='index', columns=['count'])

data.sort\_values(by='count', ascending=True, inplace=True)

Esse Dataframe por si só já é uma boa informação, já que já é possível ver os motivos que mais levam as pessoas a escolher o Windows, mas é possível melhorar ainda mais, com a demonstração visual dessa informação

Para isso, no próximo bloco de código o gráfico é criado usando o Dataframe gerado e o matplotlib, com algumas customizações para melhor aparência.

#resetando os parametros do matplotlib para o padrão e adicionando estilo

mpl.rcParams.update(mpl.rcParamsDefault)

plt.style.use('fivethirtyeight')

#criando a figura e os eixos, com tamanho de 12 por 4

fig, ax = plt.subplots(figsize=(12, 4))

#criando o gráfico de barras horizontal, com o y: opções do questionário, acessando o index do dataframe criado

#e o x: a quantidade que cada uma das opções apareceu, acessando a coluna 'count'

ax.barh(data.index, data['count'])

#colocando tamanho da fonte, titulo, label do eixo x e mudando tamanho da fonte das opções do questionario

ax.set\_title('Motivos pelos quais as pessoas escolheram o Windows')

ax.set\_xlabel('Aparições de cada resposta', fontsize=10)

ax.set\_yticks(data.index)

ax.set\_yticklabels(data.index, fontsize=8)

#usando um loop for para inserir um texto com o número de respostas 0,7 pixels após o fim do gráfico de barras

for i, v in enumerate(data['count']):

    ax.text(v+.7, i, str(v), color='black', fontsize=10, va='center')

#metodo que posiciona as coisas do melhor jeito na figura

plt.tight\_layout()

#mostrar o gráfico

plt.show()

O gráfico gerado é o seguinte:

Gráfico, Gráfico de barras

Descrição gerada automaticamente

Com esse gráfico é mais fácil de visualizar as características que influenciaram a compra de um computador com Windows, em que a principal foi, com uma grande distância para a segunda, a facilidade de uso, com 70 escolhas. E além disso é possível ver que para 23 pessoas, nenhuma característica das mostradas influenciou na compra.

A seguir, será extraído dos dados informações sobre as idades das pessoas que utilizam Windows, em comparação com os usuários de outros Sistemas operacionais também.

Uma medida interessante de utilizar seria a mediana para verificar perto de qual idade a maioria das pessoas que responderam possui, sem possuir possíveis influências de idades muito acima ou muito abaixo da central, como ocorreria na média. Mas os dados vieram como intervalos e não como a idade de cada um em si, então será usado para cada intervalo o ponto médio, como no caso de 20-30, será usado 25 para representar. Mesmo que não seja exatamente preciso, já será possível extrair uma informação interessante. Nesse caso a média também será extraída e deve ser uma informação mais interessante, porque a mediana será uma de poucas opções, sem ter como saber se os a maioria das idades são maiores ou menores que esse ponto médio que será usado para os intervalos.

Como o intervalo de menor valor possui 20 idades nele, ao invés de usar o ponto médio que seria 10, será usado 15, já que sabemos que o público ao qual o questionário foi direcionado não era infantil e, no geral, não é esperado resposta de pessoas com menos de 10 anos. E para os valores acima de 60, será utilizado 65, considerando que não é possível ter uma idade mais precisa e o limite possível é muito alto, ainda mais sabendo que foram poucas respostas em que essa opção foi marcada, como mostrado no próximo bloco de código, então se fosse muito mais alta distorceria a média, sem trazer uma informação relevante.

# seleciona apenas as linhas em que a coluna idade é igual a 'Acima de 60 anos'

# e verifica o tamanho do Dataframes Pandas resultante e imprime no console

print(len(dados[dados['idade']=='Acima de 60 anos']))

# imprime no console o tamanho do DataFrame completo

print(len(dados))

O resultado desse bloco de código é 2 e 149, podendo concluir que não é uma resposta que aparece muitas vezes no conjunto de dados.

Para calcular a média e mediana nesse bloco de código os intervalos foram transformados em um só número:

# separação dos dados em dois DataFrames, com um contendo as pessoas que responderam windows

# e o outro as pessoas que responderam qualquer coisa que não fosse windows

idade\_windows = dados[dados['SO computador']=='Windows']['idade'].reset\_index(drop=True)

idade\_naoWindows = dados[dados['SO computador']!='Windows']['idade'].reset\_index(drop=True)

# loop for para passar por todos os elementos da Pandas Series e colocar o ponto médio como valor para cada intervalo

# A ideia era usar switch case mas no Python não tem, então foi utilizada uma estrutura com ifs para atruibuir os valores

for i in range(len(idade\_windows)):

    if idade\_windows[i]=='Abaixo de 20 anos':

        idade\_windows[i] = 15

    elif idade\_windows[i]=='Entre 20 e 30 anos':

        idade\_windows[i] = 25

    elif idade\_windows[i]=='Entre 30 e 40 anos':

        idade\_windows[i] = 35

    elif idade\_windows[i]=='Entre 40 e 50 anos':

        idade\_windows[i] = 45

    elif idade\_windows[i]=='Entre 50 e 60 anos':

        idade\_windows[i] = 55

    elif idade\_windows[i]=='Acima de 60 anos':

        idade\_windows[i] = 65

for i in range(len(idade\_naoWindows)):

    if idade\_naoWindows[i]=='Abaixo de 20 anos':

        idade\_naoWindows[i] = 15

    elif idade\_naoWindows[i]=='Entre 20 e 30 anos':

        idade\_naoWindows[i] = 25

    elif idade\_naoWindows[i]=='Entre 30 e 40 anos':

        idade\_naoWindows[i] = 35

    elif idade\_naoWindows[i]=='Entre 40 e 50 anos':

        idade\_naoWindows[i] = 45

    elif idade\_naoWindows[i]=='Entre 50 e 60 anos':

        idade\_naoWindows[i] = 55

    elif idade\_naoWindows[i]=='Acima de 60 anos':

        idade\_naoWindows[i] = 65

#calculando as medias e medianas usando os métodos disponíveis no Pandas

mediana\_naowindows = idade\_naoWindows.median()

media\_naowindows = idade\_naoWindows.mean()

mediana\_windows = idade\_windows.median()

media\_windows = idade\_windows.mean()

Fazendo um DataFrame com essas médias e medianas, extraímos a tabela a seguir:

#transformando as medianas e medias em um dicionario python para fazer um dataframe no Pandas para visualizar os dados

data = {'Não Windows': [mediana\_naowindows, media\_naowindows],

        'Windows': [mediana\_windows, media\_windows]}

#tranformação do dicionário em DataFrame

mediamediana = pd.DataFrame(data, index=['Mediana', 'Média'])

print(mediamediana)

Texto

Descrição gerada automaticamente com confiança baixa

Depois disso é possível fazer alguns gráficos interessantes, com incluindo a mediana ou não necessariamente, a seguir serão feitos gráficos usando a média e a mediana, comparando as dos grupos e outros gráficos também que não usam.

A primeira ideia foi fazer um histograma, mas usando a função de gráfico de barras do matplotlib para poder comportar os dois grupos no mesmo gráfico.

import numpy as np

mpl.rcParams.update(mpl.rcParamsDefault)

plt.style.use('Solarize\_Light2')

#os intervalos a serem considerados no histograma do numpy

bins = [0, 20, 30, 40, 50, 60, 70]

bin\_labels = ['0-20', '20-30', '30-40', '40-50', '50-60', '60+']

#criando um objeto em numpy para conseguir fazer o histograma com dois valores em um só gráfico

windows\_counts, \_ = np.histogram(idade\_windows, bins=bins)

nonwindows\_counts, \_ = np.histogram(idade\_naoWindows, bins=bins)

fig, ax = plt.subplots(figsize=(10,4))

# criando o gráfico com barras e mexendo na largura para que em cada intervalo de

# idades apareçam os usuários de windows e os de outros SOs

ax.bar(bin\_labels, windows\_counts, width=-.5, align='edge', label='Usuários Windows', color='blue')

ax.bar(bin\_labels, nonwindows\_counts, width=0.5, align='edge', label='Usuários não Windows', color='green')

# colocando a legenda, que vai usar o label dos ax.bar como referencia e colocando o titulo e textos

ax.legend()

ax.set\_xlabel('Grupos de idades')

ax.set\_ylabel('Quantidade')

ax.set\_title('Quantidade das usuários Windows e de outros SOs por idade')

plt.show()

Gráfico

Descrição gerada automaticamente

Esse gráfico é gerado, mas é um pouco mais difícil de extrair o que ele quer mostrar, , já que foram muitos menos usuários não Windows no questionário, então comparar o número bruto cria uma dificuldade maior, apesar de ser um gráfico útil também, já que pode ser utilizado pra comparar a quantidade bruta e a distribuição de uma forma mais isolada pra cada um dos grupos.

Mas para poder comparar a distribuição entre os dois grupos, foi utilizado a comparação da porcentagem que cada faixa de idades representa em cada um dos grupos.

mpl.rcParams.update(mpl.rcParamsDefault)

plt.style.use('Solarize\_Light2')

bins = [10, 20, 30, 40, 50, 60, 70]

bin\_labels = ['10-20', '20-30', '30-40', '40-50', '50-60', '60+']

windows\_counts, \_ = np.histogram(idade\_windows, bins=bins)

nonwindows\_counts, \_ = np.histogram(idade\_naoWindows, bins=bins)

#nessa parte criando um dataframe para poder puxar os valores da quantidade de usuários por

# intervalo para poder usá-los em operações para descobrir a porcentagem do total

df = pd.DataFrame({'Windows': windows\_counts, 'Não Windows': nonwindows\_counts})

# loop que acessa o index do dataframe criado anteriormente na ordem

# dos grupos e insere a porcentagem calculada de cada um deles

for i, j in enumerate(bin\_labels):

    df.loc[i, 'Windows\_pct'] = (windows\_counts[i]/idade\_windows.count())\*100

    df.loc[i, 'Não Windows\_pct'] = (nonwindows\_counts[i]/idade\_naoWindows.count())\*100

fig, ax = plt.subplots(figsize=(10,4))

#mesma coisa do outro gráfico, porém dessa vez colocando como o y a coluna do dataframe diretamente

ax.bar(bin\_labels, df['Windows\_pct'], width=-.5, align='edge', label='Usuários Windows')

ax.bar(bin\_labels, df['Não Windows\_pct'], width=0.5, align='edge', label='Usuários não Windows')

#escolhendo os 'ticks' de y para poder no ax.set\_yticklabels colocar como porcentagem

ax.set\_yticks([0, 10, 20, 30, 40, 50, 60])

ax.set\_yticklabels(['0%', '10%', '20%', '30%', '40%', '50%', '60%'])

ax.set\_xlabel('Grupos de idades')

ax.set\_ylabel('Porcentagem')

ax.set\_title('Porcentagem dos usuários Windows e de outros SOs por idade(Porcentagem dos usuários)', fontsize=12)

ax.legend()

plt.tight\_layout()

plt.show()

Gráfico

Descrição gerada automaticamente

Nele é possível visualizar a diferença da representação das idades em cada gráfico, de cara dando para ver que para o Windows, uma proporção maior dos usuários que responderam é mais nova, enquanto na faixa de 20 a 30 é uma proporção maior nos outros SO do que a do Windows na mesma faixa.

E só para fazer uma visualização melhor dos dados da tabela de médias e mediana, foi feito um gráfico horizontal comparando as médias e medianas dos dois grupos.

#Extraindo as medias e medianas em variaveis separadas, para poder extrair o index e valores, tranformá-los em

# uma array do NumPy e poder trabalhar com elas no gráfico de forma mais fácil

medianas = mediamediana.loc['Mediana']

medias = mediamediana.loc['Média']

medianas\_val = medianas.to\_numpy()

medias\_val = medias.to\_numpy()

index\_medianas = medianas.index.to\_numpy()

index\_medias = medias.index.to\_numpy()

mpl.rcParams.update(mpl.rcParamsDefault)

plt.style.use('ggplot')

#criando a figura com 2 axles, de 2 linhas, com tamanho de 12 por 4, e compartilhando o x

fig, (ax1, ax2) = plt.subplots(ncols=1, nrows=2, figsize=(12, 4), sharex=True)

#construção dos dois axles do gráfico, com duas barras horizontais em cada, usando as arrays do Numpy

# e mudando as cores, já que são variáveis diferentes.

ax1.barh(index\_medianas, medianas\_val, color=['tab:purple', 'tab:pink'])

ax2.barh(index\_medias, medias\_val, color=['tab:purple', 'tab:pink'])

#inserindo texto com o valor de cada barra para uma melhor visualização, colocando 0,2 pixels a frente da barra,

# centralizada verticalmente no meio da barra

for i, v in enumerate(medianas\_val):

    ax1.text(v + 0.2, i, str(round(v, 2)), color='black', fontweight='bold', fontsize=10, va='center')

for i, v in enumerate(medias\_val):

    ax2.text(v + 0.2, i, str(round(v, 2)), color='black', fontweight='bold', fontsize=10, va='center')

#colocando titulo, labels do eixo x

ax1.set\_title('Idade dos usuários')

ax1.set\_xlabel('Mediana das idades', fontsize=10)

ax2.set\_xlabel('Média das idades', fontsize=10)

#metodo que posiciona as coisas do melhor jeito na figura

plt.tight\_layout()

#mostrar o gráfico

plt.show()

Interface gráfica do usuário

Descrição gerada automaticamente com confiança baixa

Outra informação interessante de se extrair é o sistema operacional do celular dos usuários de Windows.

Será usado o gráfico de pizza para visualizar as porcentagens de cada sistema operacional de forma mais fácil

Primeiro os dados são preparados fazendo separação e contagem deles:

#selecioanndo apenas a coluna do sistema operacional do celular das respostas

# de quem tem Windows como SO do computador e resetando o index

so\_celulares = dados[dados['SO computador']=='Windows']['SO celular'].reset\_index(drop=True)

#selecionando os valores únicos disponíveis na coluna

sos\_disp = so\_celulares.unique()

#fazendo um contador que passa por todos os valores unicos, faz uma operação lógica para

# retornar apenas as linhas do valor único e usa o método count para obter a contagem

counter = {}

for sos in sos\_disp:

    counter[sos] = so\_celulares[so\_celulares==sos].count()

Para depois montar o gráfico:

mpl.rcParams.update(mpl.rcParamsDefault)

plt.style.use('seaborn')

fig, ax = plt.subplots(figsize=(16, 5))

# Criando o gráfico de pizza, passando como parâmetro os valores disponíveis no dicinário Python,

# junto com a formatação do percentual que aparece dentro do gráfico e da distância delas do centro,

# além de mudar a cor dessas porcentagens para branco

ax.pie(counter.values(), autopct='%1.2f%%', pctdistance=.8, textprops={'color': 'white'})

#colocando titulo e legenda

ax.set\_title('Sistema operacional dos usuários Windows')

ax.legend(counter.keys())

#metodo que posiciona as coisas do melhor jeito na figura

plt.tight\_layout()

#mostrar o gráfico

plt.show()

Gráfico, Gráfico de pizza

Descrição gerada automaticamente

E para o outro gráfico que será feito, será possível reaproveitar esse bloco de código quase como um todo, fazendo as mesmas coisas, mas para outra coluna.

Será usado a coluna de escolaridade para verificar a porcentagem de cada nível de escolaridade nos usuários de Windows na pesquisa.

O código é basicamente o mesmo com algumas mudanças apenas:

escolaridade = dados[dados['SO computador']=='Windows']['escolaridade'].reset\_index(drop=True)

esc\_disp = escolaridade.unique()

counter\_esc = {}

for esc in esc\_disp:

    counter\_esc[esc] = so\_celulares[escolaridade==esc].count()

E para montar o gráfico:

mpl.rcParams.update(mpl.rcParamsDefault)

plt.style.use('seaborn')

fig, ax = plt.subplots(figsize=(15, 7))

#Importação de um mapa de cores, para selecionar as cores de forma automática

cmap = plt.cm.get\_cmap('tab20')

#uso de uma lista e um loop for para verificar o tamanho do dicionário e adicionar o número de cores correspondente

colors=[]

for i in range(len(counter\_esc)):

    colors.append(cmap(i))

# criação do gráfico de pizza, com a formatação da porcentagem, a escolha das cores

# que vem da lista criada

ax.pie(counter\_esc.values(),

       autopct='%1.2f%%',

       pctdistance=.8,

       textprops={'color': 'k'},

       colors=colors)

#colocando titulo e legenda

ax.set\_title('Escolaridade dos usuários Windows')

ax.legend(counter\_esc.keys(), bbox\_to\_anchor=(1, .9))

#metodo que posiciona as coisas do melhor jeito na figura

plt.tight\_layout()

#mostrar o gráfico

plt.show()

Gráfico, Gráfico de pizza

Descrição gerada automaticamente

Para finalizar alguns dos gráficos possíveis para visualizar as informações mais pertinentes, será feito um gráfico verificando a porcentagem que o Windows representa nas respostas, novamente usando o gráfico de pizza, que é muito eficiente em permitir a visualização das porcentagens que cada elemento dele representa. E para construí-lo, os passos serão basicamente os mesmos dos anteriores também.

sistema = dados['SO computador']

sistemas = sistema.unique()

counter\_sys = {}

for sist in sistemas:

    counter\_sys[sist] = sistema[sistema==sist].count()

counter\_sys = pd.Series(counter\_sys)

mpl.rcParams.update(mpl.rcParamsDefault)

plt.style.use('seaborn')

fig, ax = plt.subplots(figsize=(15, 7))

#Importação de um mapa de cores, para selecionar as cores de forma automática

cmap = plt.cm.get\_cmap('Dark2')

#uso de uma lista e um loop for para verificar o tamanho do dicionário e adicionar o número de cores correspondente

colors=[]

for i in range(len(counter\_sys)):

    colors.append(cmap(i))

# criação do gráfico de pizza, com a formatação da porcentagem, a escolha das cores

# que vem da lista criada

ax.pie(counter\_sys,

       autopct='%1.2f%%',

       pctdistance=.8,

       textprops={'color': 'k'},

       colors=colors)

#colocando titulo e legenda

ax.set\_title('Escolaridade dos usuários Windows')

ax.legend(counter\_sys.index, bbox\_to\_anchor=(1, .9))

#metodo que posiciona as coisas do melhor jeito na figura

plt.tight\_layout()

#mostrar o gráfico

plt.show()

Gráfico, Gráfico de pizza

Descrição gerada automaticamente

A visualização de todos essas informações fez com que os dados fossem ainda melhor aproveitados, e foram capazes de adicionar um bom conhecimento em relação as características do grupo que respondeu o questionário.

Link do código .py: https://github.com/RafaelCorreaViana/Analise-dados-ICC/blob/main/graficos.py

Link do código notebook .ipynb para melhor visualização: https://drive.google.com/file/d/1fNWkRGDUywykT9qKsmrElHeY\_25I85r4/view?usp=sharing