**AULA 1 – Automação de sistemas e processos**

Lógica de programação para resolver o desafio

Montar um passo a passo de como resolver o desafio (em portugês)

Buscar sempre reaproveitar bibliotecas já criadas (base de dados – pandas)

**# comentários**

**!pip install biblioteca**

Bibliotecas

**pyautogui – automatiza cliques e teclas para interagir com a aplicação**

**pyperclip – automatiza tela do computador para copiar e colar informações**

**time – atribui um valor de espera ao programa**

**pandas – biblioteca para trabalhar com base de dados**

**pyautogui.PAUSE = 1 – pausa que vai fazer entre um comando e outro do autogui (1 segundos), adicionar um delay**

**import pyautogui**

**import pyperclip**

**pyautogui.hotkey(“ctrl”,”t”)/.press/.click – usar atalhos,está abrindo uma aba no google**

**pyautogui.write(“https://drive.google.com/...?”) – escreve o link no navegador, porém sem os caracteres especiais**

**pyperclip.copy(“https://drive.google.com/...?”) – copia o link inteiro, sem exclusão e caracteres**

**pyautogui.hotkey(“ctrl”,”v”) – colar o link copiado**

**pyautogui.press(“enter”)**

**time.sleep(5) – delay de 5 segundos, importante colocar caso o tempo de carregamento entre uma página e outra possa demorar para abrir**

Dentro do gmail

**Pyautogui.click(x=974,y=676, clicks = 2)**

**pyautogui.position() – pegar a posição atual do mouse**

Análise de dados

**import pandas**

**table = pandas.read\_excel(r“C:\Desktop\Python\Hashtag\Aula 1\Vendas – Dez.xlsx”) – o r antes do diretório diz que não é para tentar ler caracteres especiais**

**display(table) – exibe a tabela, igual ao print porém mais apresentável**

Calcular o faturamento a partir dos dados

**Faturamento = table[“Valor Final”].sum()**

**Qde\_produtos = table[“Quantidade”].sum()**

Escrever o email

**pyperclip.copy(“Relatório de Vendas”) – há um caracter especial (acento agudo)**

**pyautogui.hotkey(“ctrl”,”v”)**

**Text = f“””**

**Prezados, bom dia**

**O faturamento foi de: R$ {Faturamento:,.2f}**

**A quantidade de produtos foi de: {Qde\_produtos:}**

**“”” – o caracter especial f é usado para indicar que vamos formatar alguma informação, a “,” indica separador de milhar, o “.2f” indica duas casas decimais**

**Pyperclip.copy(Text)**

**Pyautogui.hotkey(“ctrl”,”v”)**

**AULA 2 – Análise de dados**

Kaggle – site usado para pegar base de dados para treino, entre outros

Colunas que não apresentam informações relevantes, deve-se excluir para não pesar a base de dados

**!pip install plotly**

**import pandas as pd**

**import plotly.express as px**

Importar dados

**Tabela = pd.read\_csv(“telecom\_users.csv”)**

Dropando a coluna sem dados coerentes para análise

**Tabela = Tabela.drop(“Unnamed: 0”, axis = 1) – axis = 1 é a coluna. Se fosse deletar a linha, ou índice: Tabela.drop(“0”, axis = 0)**

Utilizando

**print(Tabela.info()), observa-se que os dados de TotalGasto estão em object (texto) e não em float**

**tabela[“TotalGasto”] = pd.to\_numeric(tabela[“TotalGasto”], errors = “coerce”) – o erro coerce irá transformar qualquer valor que seja não numérico, tipo algum preenchimento errado, em valor númerico ou NaN**

Tratando valores vazios(Havia uma coluna NaN e algumas poucas linhas com informações NaN)

**Tabela = Tabela.dropna(how = “all”, axis = 1) – excluir as colunas vazias NaN**

**Tabela = Tabela.dropna(how = “any”, axis = 0) – excluir as linhas com no mínimo um valor vazio NaN**

Análise Inicial

**Tabela[“Churn”].value\_counts(normalize = True) – Conta os valores apresentados e normaliza**

**Tabela[“Churn”].value\_counts(normalize = True).map(“{:.1%}”.format) – Adicionado código de formatação, a partir de map, para gerar valores percentuais**

Análise detalhada

**for item in Tabela.columns: - para cada item referente a coluna da tabela**

**grafico = px.histogram(Tabela, x=item, color=”Churn”)**

**grafico.show()**

**AULA 3 – Automação WEB e Busca de Informações com Python**

Conseguir pesquisar as cotações de Dólar, Euro e Ouro para atualizar o valor de certos produtos

**Selenium** – permite controlar o navegador, bom para automações na internet

**!pip install selenium**

Webdriver para instalar – chromedriver - comunica o navegador com o selenium

Onde colocar o chromedriver? Está na apostila, colocar no lugar onde está o executável do python

**from selenium import webdriver – importa funções específicas da biblioteca**

**from selenium.webdriver.common.keys import Keys**

**from selenium.webdriver.common.by import By**

**navegador = webdriver.Chrome()**

**navegador = webdriver.Chrome(“chromedriver.exe”) – quando o .exe estiver no mesmo local do arquivo**

Navegando no Google para pegar as cotações

**navegador.get(“https://www.google.com/”)**

**navegador.find\_element(By.XPATH,’** **/html/body/div[1]/div[3]/form/div[1]/div[1]/div[1]/div[2]/div[2]/input’).send\_keys(“cotação dólar”) – entre aspas coloca o XPATH do elemento na tela do navegador, para encontrar o mesmo utiliza-se a opção inspecionar, clicar no “mouse”, selecionar e copiar o XPATH do elemento, o XPATH no selenium deve-se usar aspas simples**

**navegador.find\_element(By.XPATH,’** **/html/body/div[1]/div[3]/form/div[1]/div[1]/div[1]/div[2]/div[2]/input’).send\_keys(Keys.ENTER) – iniciar a pesquisa após preencher a barra de pesquisa**

**cotacao\_dolar = navegador.find\_element(By.XPATH,’** **//\*[@id="knowledge-currency\_\_updatable-data-column"]/div[1]/div[2]/span[1]’).get\_attribute(‘data-value’) – Pega algum atributo da tag do elemento**

Tratando valores com separador decimal “vírgula” para “ponto”

**cotacao\_ouro = cotacao\_ouro.replace(“,”,”.”) - o valor estava como 320,88 (o padrão do Python é 320.88)**

Importar a base de dados e atualizar as cotações na base

**import pandas as pd**

**tabela = pd.read\_excel(“base de dados”)**

**tabela.loc[linha, coluna] = algum\_valor**

**tabela.loc[tabela[‘Moeda’] == ‘Dólar’,’Cotação’] = float(cotacao\_dolar) – utiliza o float pq o valor que se pega no navegador é uma string. Poderia utilizar um for e if else, mas usando as ferramentas do Pandas fica mais eficiente (o for para dólar, euro e ouro)**

Atualizando o preço de compra e depois o de venda: Compra = Preço Original\*Cotação, Venda = Preço Compra\*Margem de lucro

**tabela[“Preço de Venda”] = tabela[“Preço de Compra”]\*tabela[“Margem”]**

Exportar o dataframe

**tabela.to\_excel(“Produtos\_Novo.xlsx”, index = False) – False faz com que não seja passado os valores das linhas**

**AULA 4 – Ciência de Dados e Inteligência Artificial**

Há uma base de dados para importar para treinar a IA  
Análise exploratória – basicamente tentaremos visualizar a correlação entre itens que estão distribuídos e identificar como as informações estão se comportando entre si

Correlação – se a propaganda na TV dobra, e as vendas dobram, há correlação próxima de 1

Gráficos no python – plotly, matplotlib e seaborn

Inteligência Artificial – deve-se separa um conjunto de dados de testes para treinar a inteligência. Y é o objetivo e X são os dados de treino para previsão de Y

Gerenciador de IA’s – scikit-learn, tensorflow, keras, opencv, neat

**.info – ajuda a analisar o dataframe e identifica algum dado incoerente**

**!pip install matplotlib – apresenta melhor integração e exibição de gráficos**

**!pip install seaborn – extensão do matplotlib com gráficos mais apresentáveis**

**!pip install scikit-learn – gerenciador da inteligência artificial**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

Verificando a correlação dos dados por gráficos

**sns.heatmap(table.corr(), cmap=”Wistia”, annot=True) – printa a correlação entre os índices, pegando como base os dados do dataframe**

**plt.show()**

Treino da IA e previsão (por enquanto é somente treino, após isso será feito os testes)

**y = table[“Vendas”] – o que será previsto**

**x = table[[“TV”,”Radio”,”Jornal”]] – dados para previsão**

**from sklearn.model\_selection import train\_test\_split**

**x\_treino, x\_teste, y\_treino,y\_teste = train\_test\_split(x,y, test\_size=0.2, random\_state=1) – o tts separa 80% dos dados para treino e 20% para teste. O random\_state é para que durante o treino o mesmo conjunto de dados sejam utilizados para teste, sem alterá-los**

Criação da IA – Modelos existentes (Redes neurais, regressão linear, árvore de decisão)

Regressão linear – traçar uma reta

Árvore de decisão – teste com perguntas e respostas (sim, não)

**from sklearn.linear\_model import LinearRegresseion**

**from sklearn.linear\_model import RandomForestRegressor**

**modelo\_regressaolinear = LinearRegression()**

**modelo\_arvoredecisao = RandomForestRegressor()**

Treinamento (os 80%)

**modelo\_regressaolinear.fit(x\_treino,y\_treino)**

**modelo\_arvoredecisao.fit(x\_treino,y\_treino)**

Teste (os 20%)

**previsao\_regressaolinear = modelo\_regressaolinear.predict(x\_teste)**

**previsao\_arvoredecisao = modelo\_arvoredecisao.predict(x\_teste)**

Comparação utilizando o R² (utiliza-se o y\_teste para comparar com as previsões estimadas)

**from sklearn import metrics**

**print(metrics.r2\_score(y\_teste, previsao\_regressaolinear).map(“{:.1%}”.format))**

**print(metrics.r2\_score(y\_teste, previsao\_arvoredecisao).map(“{:.1%}”.format))**

Previsão de novos valores

**new = pd.read\_csv(“novos.csv”)**

**display(novos)**

**previsao = modelo\_arvoredecisao.predict(novos)**

**print(previsao)**

DÚVIDAS

1. Como transformar em executável?