

Tutorial mineração de dados aplicada ao ENADE

Requisitos:

- Visual Studio Code
- Jupyter Notebook
- Python 3.7.5

Primeiramente é preciso baixar os códigos do seguinte repositório:

https://github.com/destandroid/TCC_ENADE2021

Instalação de bibliotecas:

```
# bibliotecas necessárias
!pip install pandas matplotlib seaborn unicode numpy
```

Tratamento dos dados:

Primeiramente abrir o arquivo *Tratamento_vetores_e_auxiliar.ipynb* no Visual Studio Code, lembrando que a pasta dos microdados do ENADE precisa estar no mesmo diretório, inicialmente é necessário executar o código que vai criar as pastas, onde vão ser armazenadas as análises do ENADE.

No Visual Studio Code é possível executar o código com o símbolo com a cor amarela, isso vai executar todos os códigos, começando nesse ate a ultima celula de código no arquivo



```
import os

#pasta aux_files
aux_files = "aux_files"
try:
    os.makedirs(aux_files)
    print(f"Pasta '{aux_files}' criada com sucesso.")
except FileExistsError:
    print(f"A pasta '{aux_files}' já existe.")

#pasta microdados_tratados
microdados_tratados = "microdados_tratados"

try:
    os.makedirs(microdados_tratados)
    print(f"Pasta '{microdados_tratados}' criada com sucesso.")
except FileExistsError:
    print(f"A pasta '{microdados_tratados}' já existe.")
```

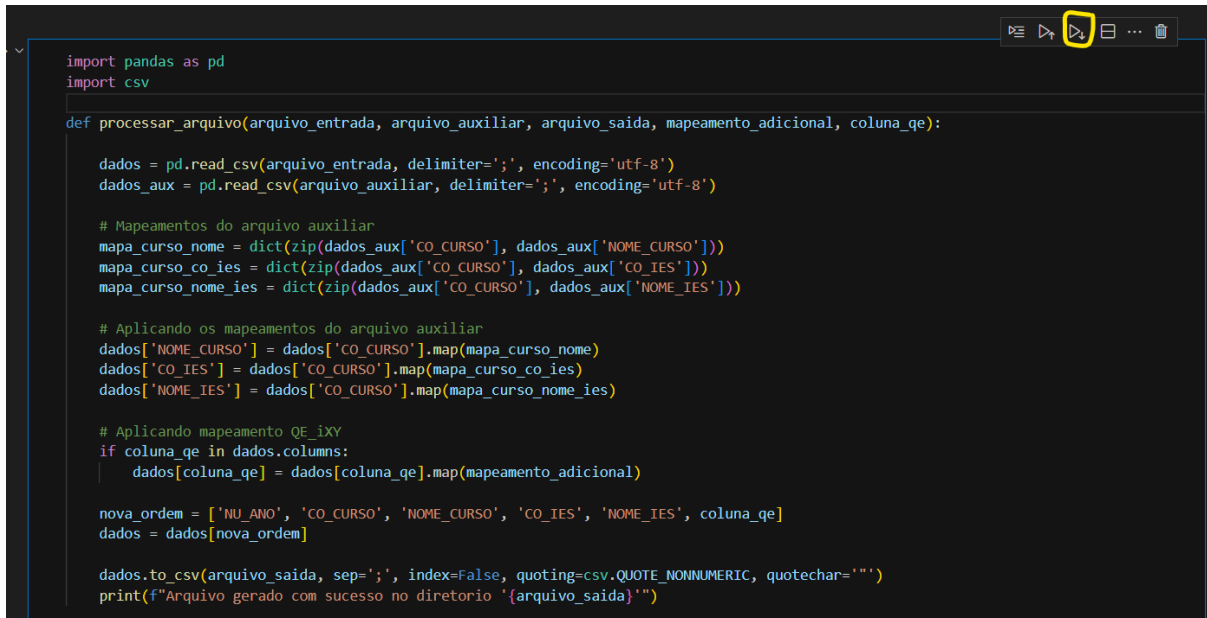
Depois de executar todos os códigos vão ser gerados os seguintes arquivos nos seguintes diretórios:

- aux_files/vetor_microdado_3.csv
- aux_files/microdados_aux.csv
- microdados_tratados/form_microdados_arq3_vetores_filtrados.csv

- microdados_tratados/form_microdados_arq3_percepcao_prova.csv

O segundo tratamento de arquivos é feito no arquivo *Tratamento_microdados.ipynb*, onde são feitos os mapeamentos para os arquivos que contêm dados do questionário do estudante.

Novamente aperte o botão em destaque do círculo amarelo para executar todas as células de código.



```
import pandas as pd
import csv

def processar_arquivo(arquivo_entrada, arquivo_auxiliar, arquivo_saida, mapeamento_adicional, coluna_qe):

    dados = pd.read_csv(arquivo_entrada, delimiter=';', encoding='utf-8')
    dados_aux = pd.read_csv(arquivo_auxiliar, delimiter=';', encoding='utf-8')

    # Mapeamentos do arquivo auxiliar
    mapa_curso_nome = dict(zip(dados_aux['CO_CURSO'], dados_aux['NOME_CURSO']))
    mapa_curso_co_ies = dict(zip(dados_aux['CO_CURSO'], dados_aux['CO_IES']))
    mapa_curso_nome_ies = dict(zip(dados_aux['CO_CURSO'], dados_aux['NOME_IES']))

    # Aplicando os mapeamentos do arquivo auxiliar
    dados['NOME_CURSO'] = dados['CO_CURSO'].map(mapa_curso_nome)
    dados['CO_IES'] = dados['CO_CURSO'].map(mapa_curso_co_ies)
    dados['NOME_IES'] = dados['CO_CURSO'].map(mapa_curso_nome_ies)

    # Aplicando mapeamento QE_iXY
    if coluna_qe in dados.columns:
        dados[coluna_qe] = dados[coluna_qe].map(mapeamento_adicional)

    nova_ordem = ['NU_ANO', 'CO_CURSO', 'NOME_CURSO', 'CO_IES', 'NOME_IES', coluna_qe]
    dados = dados[nova_ordem]

    dados.to_csv(arquivo_saida, sep=';', index=False, quoting=csv.QUOTE_NONNUMERIC, quotechar='')
    print(f"Arquivo gerado com sucesso no diretorio '{arquivo_saida}'")
```

Cada arquivo para o tratamento tem a seguinte estrutura:



```
microdados2021_arq5

Edição, código de curso e sexo

mapeamento_tp_sexo = {
    "M": "Masculino",
    "F": "Feminino"
}

processar_arquivo(
    'microdados_Enade_2021_LGPD/2.DADOS/microdados2021_arq5.txt', #arquivo_entrada
    'aux_files/vetores/microdados_aux.csv', #arquivo_auxiliar para o mapeamento do CO_CURSO
    f'{dir_destino}/form_microdados_arq5.csv', #arquivo_saida
    mapeamento_tp_sexo, #mapeamento
    'TP_SEXO' #cabecalho coluna
)
```

A estrutura depende do conteúdo de cada arquivo, é preciso olhar o *Dicionário_arquivos_variáveis_microdados_Enade_2021* para ter um melhor conhecimento do conteúdo de cada arquivo.

Depois de executar todos os códigos vão ser gerados os seguintes arquivos nos seguintes diretórios:

- microdados_tratados/form_microdados_arq4.csv até o arquivo
- microdados_tratados/form_microdados_arq32.csv
- microdados_tratados/form_microdados_arq43.csv

Análise dos dados:

Para começar a gerar as análises, precisamos abrir o arquivo *Analise_dados_enade.ipynb*, novamente precisamos criar algumas pastas específicas para os arquivos.

Também é possível executar cada célula de forma individual apertando o play na parte de cima esquerda, está destacado com a cor amarela.



```
import os

#pasta tabelas
diretorio = "tabelas"
try:
    os.makedirs(diretorio)
    print(f"Pasta '{diretorio}' criada com sucesso.")
except FileExistsError:
    print(f"A pasta '{diretorio}' já existe.")

#pasta tabelas/vetores
vetores = f"{diretorio}/vetores"
try:
    os.makedirs(vetores)
    print(f"Pasta '{vetores}' criada com sucesso.")
except FileExistsError:
    print(f"A pasta '{vetores}' já existe.")

#pasta tabelas/questionario
questionario = f"{diretorio}/questionario"
try:
    os.makedirs(questionario)
    print(f"Pasta '{questionario}' criada com sucesso.")
except FileExistsError:
    print(f"A pasta '{questionario}' já existe.")

#pasta tabelas/enade
enade = f"{diretorio}/enade"
try:
    os.makedirs(enade)
    print(f"Pasta '{enade}' criada com sucesso.")
except FileExistsError:
    print(f"A pasta '{enade}' já existe.")
```

Para cada análise o código está dividido em duas partes:

- Função que faz a análise, para fazer a análise de cada célula de forma individual é preciso apertar o play na parte de cima esquerda, está destacado com a cor amarela.

Análise de alunos

Total de alunos por curso

```
import pandas as pd
def total_alunos_por_curso(dir_entrada):
    df = pd.read_csv(dir_entrada, sep=';', usecols=['NOME_CURSO', 'NT_GER'])

    df['CURSO_ORIGINAL'] = df['NOME_CURSO']
    df['NOME_CURSO'] = df['NOME_CURSO'].str.lower()

    contagem_total = df.groupby('NOME_CURSO').size()
    contagem_validos = df[df['NT_GER'].notna() & (df['NT_GER'] != '')].groupby('NOME_CURSO').size()
    percentual_realizacao = (contagem_validos / contagem_total * 100).round(2).astype(str) + '%'

    result = pd.DataFrame({
        'Número total de alunos\ninscritos': contagem_total,
        'Número de alunos\nque realizaram a prova': contagem_validos,
        'Percentual que realizou\nna prova': percentual_realizacao
    }).reset_index()

    result = result.join(df[['NOME_CURSO', 'CURSO_ORIGINAL']].drop_duplicates().set_index('NOME_CURSO'), on='NOME_CURSO')
    result.rename(columns={'CURSO_ORIGINAL': 'Cursos participantes do ENADE'}, inplace=True)

    #soma total das colunas
    soma_total = result[['Número total de alunos\ninscritos', 'Número de alunos\nque realizaram a prova']].sum()
    soma_total['Cursos participantes do ENADE'] = 'Total alunos ENADE'

    #percentual para a soma total
    percentual_soma_total = (soma_total['Número de alunos\nque realizaram a prova'] / soma_total['Número total de alunos\ninscritos']) * 100
    soma_total['Percentual que realizou\nna prova'] = f"round(percentual soma total, 2)%"
```

- Célula com os dados de input e chamada da função.

```
dir_entrada = 'microdados_tratados/form_microdados_arq3_vetores_filtrados.csv'
dir_destino = f"{enade}/tabela_total_alunos_ENADE"

contagem_por_curso = total_alunos_por_curso(dir_entrada)
tabela_total_alunos_enade(contagem_por_curso, dir_destino)
```

Se for preciso gerar todos os análises do arquivo tem que ser utilizado o botão destacado em amarelo, que vai executar todas as células de código abaixo até percorrer o arquivo todo.

Análise dados ENADE

Análise de alunos

Total de alunos por curso

```
import pandas as pd
def total_alunos_por_curso(dir_entrada):
    df = pd.read_csv(dir_entrada, sep=';', usecols=['NOME_CURSO', 'NT_GER'])

    df['CURSO_ORIGINAL'] = df['NOME_CURSO']
    df['NOME_CURSO'] = df['NOME_CURSO'].str.lower()

    contagem_total = df.groupby('NOME_CURSO').size()
    contagem_validos = df[df['NT_GER'].notna() & (df['NT_GER'] != '')].groupby('NOME_CURSO').size()
    percentual_realizacao = (contagem_validos / contagem_total * 100).round(2).astype(str) + '%'

    result = pd.DataFrame({
        'Número total de alunos\ninscritos': contagem_total,
        'Número de alunos\nque realizaram a prova': contagem_validos,
        'Percentual que realizou\nna prova': percentual_realizacao
    }).reset_index()
```

Para as análises mais específicas é possível escolher um curso e uma faculdade específica, isso é feito na seguinte célula:

Entrada de dados analisados(Microdados)

```
#lista de cursos para a analise
cursos_analisados = ['ciencia Da Computacao (bacharelado)', 'ciencia da computacao (licenciatura)' ]
#codigo ies para a faculdade analisada
faculdades_filtradas_CO_IES=[2]
print(faculdades_filtradas_CO_IES)
```

Para *faculdades_filtradas_CO_IES* é preciso saber o código da IES, no exemplo o código 2 representa a Universidade de Brasília.

Análise de vetores:

Para a análise de vetores tem os seguintes parâmetros de entrada:

```
faculdades_analisadas_CO_IES = [575, 570, 8, 576, 581, 571, 577, 573, 585, 4925, 54, 591, 7, 56, 572, 586, 582, 271, 2, 57]
cursos_analisados = ['ciencia da computacao (licenciatura)', 'ciencia da computacao (Bacharelado)' ]

faculdade_destacada= 'UNIVERSIDADE DE BRASILIA'
```

onde:

- *faculdades_analisadas_CO_IES*: São os códigos da IES que vão ser analisados e colocados em cada tabela. Nesse exemplo cada código representa:

```
map_sigla_ies = {
    575: 'UFMG',
    570: 'UFRN',
    8: 'UFV',
    576: 'UFJF',
    581: 'UFRGS',
    571: 'UFPR',
    577: 'UFAL',
    573: 'UFES',
    585: 'UFSC',
    4925: 'UFABC',
    54: 'UNICAMP',
    591: 'UNIFESP',
    7: 'UFSCAR',
    56: 'UNESP',
    572: 'UFF',
    586: 'UFRJ',
    582: 'UFSM',
    271: 'UNOESTE',
    2: 'UNB',
    57: 'UEM'
}
```

- `cursos_analisados`: Representa os cursos que vão ser filtrados para a análise.
- `faculdade_destacada`: Representa a faculdade que vai ser destacada com uma cor verde nas tabelas, o exemplo é 'UNIVERSIDADE DE BRASÍLIA'.

Cada célula vai utilizar os parâmetros definidos para a análise.

Finalmente o arquivo *Análise Vetores.ipynb* faz a análise para as questões objetivas da formação geral e componente específico. Tendo como parâmetros de entrada os seguintes:

```
#Cursos a ser analisados
cursos_analisados = [ "Ciência Da Computação (Bacharelado)", 'ciencia da
computacao (licenciatura)']
#Faculdades pelo CO_IES as 20 melhores
faculdades_analisadas_CO_IES = [575, 570, 8, 576, 581, 571, 577, 573, 585,
4925, 54, 591, 7, 56, 572, 586, 582, 271, 2,57]

#Faculdade que vai ser destacada na tabela com a cor verde '#A4FFA4'
faculdade_destacada= 'UNIVERSIDADE DE BRASILIA'
sigla_destacada="UNB"
```

Para cada tabela é preciso escolher as questões que vão ser analisadas, no exemplo primeiramente é criada uma imagem com as questões 1 até 4 da formação geral.

```
colunas_questoes = [f"forma_geral_q_{i}" for i in range(1, 5)]
for curso_analisado in cursos_analisados:

    dir_destino = "FG1"
    percentuais_acertos, mapeamento_ies_curso =
percentual_acertos_por_faculdade(
        dir_entrada, [curso_analisado], faculdades_analisadas_CO_IES,
colunas_questoes
    )
    tabela_ranking_qe_horizontal(percentuais_acertos, mapeamento_ies_curso,
curso_analisado, dir_destino)
```

Depois é criada uma imagem com as questões 5 até 8 da formação geral.

```
colunas_questoes = [f"forma_geral_q_{i}" for i in range(5, 9)]
for curso_analisado in cursos_analisados:

    dir_destino = "FG2"
    percentuais_acertos, mapeamento_ies_curso =
percentual_acertos_por_faculdade(
        dir_entrada, [curso_analisado], faculdades_analisadas_CO_IES,
colunas_questoes
```

```
)  
    tabela_ranking_qe_horizontal(percentuais_acertos, mapeamento_ies_curso,  
curso_analisado, dir_destino)
```

Para as questões do componente específico é necessário verificar manualmente quais questões foram anuladas, para que não sejam colocadas na tabela de questões.