FACULDADE SENAC PERNAMBUCO
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Daniel Oliveira

Dayvson Lima

Natan Gonçalves

Reginaldo Alves

Luis Henrique

RELATÓRIO
OBESITY LEVELS (NÍVEIS DE OBESIDADE)

Recife,
2024

Daniel Oliveira

Dayvson Lima

Natan Gonçalves

Reginaldo Alves

Luis Henrique

RELATÓRIO
OBESITY LEVELS (NÍVEIS DE OBESIDADE)

Relatório apresentado para compor conceito do 2º Processo Avaliativo da disciplina de Data Science, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Faculdade Senac Pernambuco.

Professor Orientador: Marco Aurelio

Recife,
2024

SUMÁRIO

Introdução

A obesidade é um problema de saúde global com consequências significativas. Com o aumento dos casos de obesidade, torna-se necessário realizar novas pesquisas para examinar os fatores que influenciam essa condição e prever a ocorrência de determinadas doenças. Este relatório apresenta uma análise do conjunto de dados "Obesity Levels", que inclui 2.111 registros e 17 variáveis, sem valores nulos ou zerados. O dataset contém informações sobre indivíduos do México, Peru e Colômbia, coletadas através de uma plataforma web e geradas sinteticamente usando a ferramenta Weka e o filtro SMOTE.

Descrição do Dataset

O dataset contém as seguintes variáveis:

- **NObeyesdad**: Nível de obesidade (variável alvo)
- **Gender**: Gênero
- **Age**: Idade
- **Height**: Altura
- **Weight**: Peso
- **family_history_with_overweight**: Histórico familiar de sobrepeso
- **FAVC**: Frequência de consumo de alimentos com alto teor calórico
- **FCVC**: Frequência de consumo de vegetais
- **NCP**: Número de refeições principais diárias
- **CAEC**: Consumo de alimentos entre as refeições
- **SMOKE**: Hábito de fumar
- **CH2O**: Consumo diário de água
- **SCC**: Monitoramento de calorias ingeridas
- **FAF**: Frequência de atividade física
- **TER**: Tempo de uso de dispositivos tecnológicos
- **CALC**: Frequência de consumo de álcool
- **MTRANS**: Meio de transporte usual

Requisitos

Para garantir que todos os scripts e notebooks funcionem corretamente, é necessário instalar os seguintes pacotes Python. Abaixo está a lista de pacotes necessários e o comando para instalá-los.

Pacotes Necessários

- **matplotlib**: Biblioteca para criação de gráficos estáticos, animados e interativos.
- **seaborn**: Biblioteca baseada em matplotlib que fornece uma interface de alto nível para desenhar gráficos estatísticos atraentes e informativos.
- **scikit-learn**: Biblioteca para aprendizado de máquina que fornece ferramentas simples e eficientes para análise de dados e mineração de dados.
- **dash**: Biblioteca para construção de dashboards web interativos com Python.
- **pandas**: Biblioteca para manipulação e análise de dados, oferecendo estruturas de dados e operações para manipular tabelas numéricas e séries temporais.
- **numpy**: Biblioteca fundamental para computação científica em Python, fornecendo suporte para arrays e matrizes grandes multidimensionais, junto com uma grande coleção de funções matemáticas.

Comando de Instalação

Para instalar todas essas bibliotecas, execute o seguinte comando no terminal:

```
pip install matplotlib seaborn scikit-learn dash pandas numpy
```

Análise Exploratória dos Dados

- **Estrutura do Dataset**

```
# Lendo a base de dados utilizada.
obesityDataSet =
pd.read_csv('ObesityDataSet_raw_and_data_sinthetic.csv', sep=',')
display(obesityDataSet)

obesityDataSet.shape  # Verificando o número de registros e
variáveis
obesityDataSet.dtypes  # Checando tipos das colunas presentes
obesityDataSet.isnull().sum()  # Verificando valores nulos
obesityDataSet.nunique()  # Contando valores únicos por coluna
```

- **Verificação das Categorias da Variável Alvo**

```
# Verificando se ha alguma categoria ´NObeyesdad´ diferente das
categorias padrão.
subset =
obesityDataSet.loc[~obesityDataSet['NObeyesdad'].isin(['Normal_Wei
ght', 'Overweight_Level_I', 'Overweight_Level_II',
'Overweight_Level_III', 'Obesity_Type_III', 'Obesity_Type_II',
'Obesity_Type_I'])]

display(subset)
```

- **Análise Focada nos Casos de Obesidade**

```
# Filtrando dados para focar apenas nos casos de obesidade
obesityTypeDataSet =
obesityDataSet.loc[(obesityDataSet['NObeyesdad'] !=
'Normal_Weight') & (obesityDataSet['NObeyesdad'] !=
'Overweight_Level_I') & (obesityDataSet['NObeyesdad'] !=
'Overweight_Level_II') & (obesityDataSet['NObeyesdad'] !=
'Overweight_Level_III') & (obesityDataSet['NObeyesdad'] !=
'Insufficient_Weight')]
display(obesityTypeDataSet)
```

```
# Criando a coluna 'imc': Índice de Massa Corporal
obesityTypeDataSet.loc[:,'imc'] = (obesityTypeDataSet['Weight']) /
(obesityTypeDataSet['Height'] ** 2)
display(obesityTypeDataSet)

# Mudando o tipo das idades de Float para int
obesityTypeDataSet.loc[:, 'Age'] =
obesityTypeDataSet['Age'].astype(int)
display(obesityTypeDataSet)
```

- **Análise Descritiva por Categoria de Obesidade**

Dividimos o conjunto de dados pela categoria 'NObeyesdad', agrupando-os de acordo com o tipo de obesidade (1, 2 ou 3). Em seguida, calculamos as médias de idade para cada sexo dentro de cada categoria de obesidade.

```
# Separando dataSet por Categoria 'NObeyesdad'
obesityLevel_I =
obesityTypeDataSet.loc[obesityTypeDataSet['NObeyesdad'] ==
'Obesity_Type_I']
display(obesityLevel_I)

# Tirando a média de idade por sexo
mean_age_male_Obesity_I =
obesityLevel_I.loc[obesityLevel_I['Gender'] == 'Male',
'Age'].mean().astype(int)
mean_age_female_Obesity_I =
obesityLevel_I.loc[obesityLevel_I['Gender'] == 'Female',
'Age'].mean().astype(int)

print("Média de idade entre homens:", mean_age_male_Obesity_I)
print("Média de idade entre mulheres:", mean_age_female_Obesity_I)
```

- **Análise da média e mediana da idade por sexo**

```
# Calculando a média e a mediana da idade por sexo.
```

```
mean_obesityData_Man =
obesityTypeDataSet.loc[obesityTypeDataSet['Gender'] == 'Male',
'Age'].mean()

mean_obesityData_Woman =
obesityTypeDataSet.loc[obesityTypeDataSet['Gender'] == 'Female',
'Age'].mean()


print("Man mean",mean_obesityData_Man)

print("Woman mean",mean_obesityData_Woman)


median_man =
np.median(obesityTypeDataSet.loc[obesityTypeDataSet['Gender'] ==
'Male', 'Age'])
median_woman =
np.median(obesityTypeDataSet.loc[obesityTypeDataSet['Gender'] ==
'Female', 'Age'])


print("Man median",mean_obesityData_Man)

print("Woman median",median_woman)

# Ambos os casos mostram que a média é maior do que a mediana.
Isso sugere que pode haver alguns valores extremos de idade
# mais altos que estão puxando a média para cima em ambos os
grupos (homens e mulheres). Esses valores extremos podem
# representar uma pequena porcentagem da população, enquanto a
maioria das idades está abaixo da média.
```

- **Análise pelo consumo de alimentos e hábitos**

```
# Criando dataSete para determinada analise.
consumo_de_Alimentos_e_habitos = obesityTypeDataSet[['Age',
'Gender', 'FAVC', 'FCVC', 'NCP', 'family_history_with_overweight',
'imc', 'NObeyesdad']]

display(consumo_de_Alimentos_e_habitos)
```

```
consumo_de_Alimentos_e_habitos.shape
```

- **Análise de Correlação**

```
from sklearn.preprocessing import LabelEncoder

# Inicializando o LabelEncoder
label_encoder = LabelEncoder()
df_encoded = obesityTypeDataSet.copy()

# Colunas a serem transformadas
columns_to_encode = ['Gender', 'CALC', 'FAVC', 'SCC', 'SMOKE',
'family_history_with_overweight', 'CAEC', 'MTRANS', 'NObeyesdad']
for column in columns_to_encode:
    df_encoded[column] =
label_encoder.fit_transform(df_encoded[column])

# Calculando a matriz de correlação
correlation = df_encoded.corr()

# Plot da matriz de correlação
sns.heatmap(correlation, annot = True, fmt=".1f", linewidths=.6)
```

  - **Principais Correlações Identificadas**

    - Age & MTRANS: -0.72 (correlação forte)
      - O transporte usado tem uma forte relação com a idade do indivíduo.
    - Weight & imc: 0.77 (correlação forte)
      - O IMC do indivíduo tem uma forte relação com o seu peso.
    - Height & Weight: 0.63 (correlação moderada)
      - A altura do indivíduo tem uma relação moderada com o peso.
    - Weight & NObeyesdad: 0.67 (correlação moderada)
      - O peso do indivíduo tem uma relação moderada com o nível de obesidade.
    - FCVC & NObeyesdad: 0.65 (correlação moderada)

    - O nível de obesidade tem uma relação moderada com o consumo de vegetais.

- FCVC & imc: 0.61 (correlação moderada)
    - O IMC tem uma relação moderada com o consumo de vegetais.

- NObeyesdad & imc: 0.91 (correlação muito forte)
    - O IMC de um indivíduo tem uma relação muito forte com seu nível de obesidade.

- **Visualização Interativa com Dash**

```
from dash import Dash, html, dcc, callback, Output, Input
import plotly.express as px
import pandas as pd

# Supondo que df_encoded já está carregado e preparado
df = obesityTypeDataSet.copy()

# Certifique-se de que 'Age', 'MTRANS', 'Weight' e 'imc' estão no formato correto
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')  # Converte 'Age' para numérico, substituindo erros por NaN
df['MTRANS'] = df['MTRANS'].astype(str)  # Converte 'MTRANS' para string (se necessário)
df['Weight'] = pd.to_numeric(df['Weight'], errors='coerce')  # Converte 'Weight' para numérico
df['Weight'] = df['Weight'].fillna(0).astype(int)  # Converte 'Weight' para inteiro, preenchendo NaN com 0
df['imc'] = pd.to_numeric(df['imc'], errors='coerce')  # Converte 'imc' para numérico

# Convertendo IMC para inteiro
```

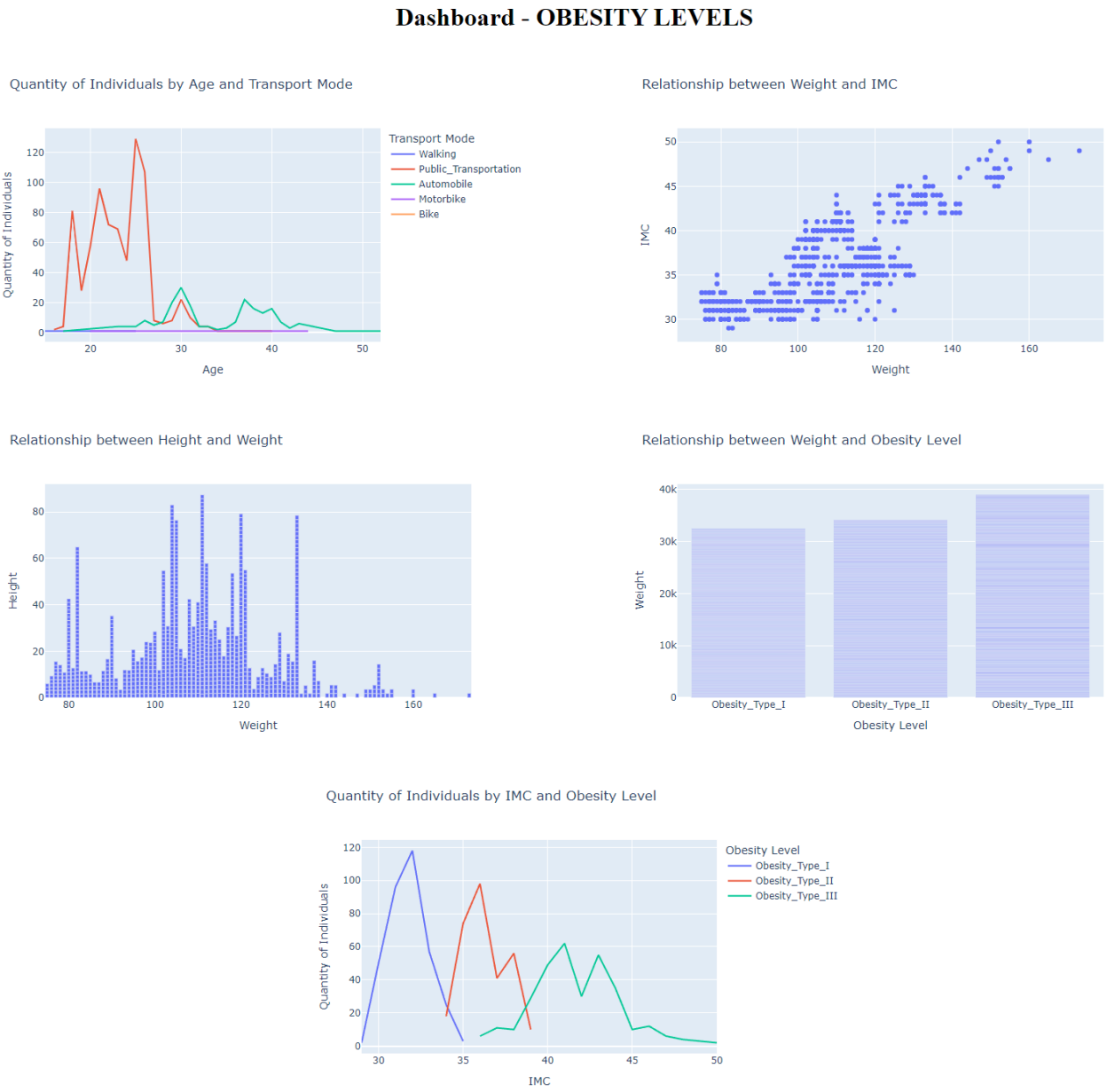
```
df['imc'] = df['imc'].fillna(0).astype(int)

# Arredonda a coluna 'Height' para duas casas decimais
df_count_hw['Height'] = df_count_hw['Height'].round(2)


# Contagem de indivíduos por Age e MTRANS
df_count = df.groupby(['Age',
'MTRANS']).size().reset_index(name='count')


# Contagem de indivíduos por imc e nobeyesdad
df_count_imc_nob = df.groupby(['imc',
'NObeyesdad']).size().reset_index(name='count')

app = Dash(__name__)

app.layout = html.Div([
    html.H1(children='Dashbord - OBESITY LEVELS',
style={'textAlign': 'center'}),
    html.Div([
        dcc.Graph(id='line-graph-age-mtrans'),
        dcc.Graph(id='scatter-graph-weight-imc')
    ], style={'display': 'flex', 'justify-content':
'space-around'}),
    html.Div([
        dcc.Graph(id='bar-graph-height-weight'),
        dcc.Graph(id='bar-graph-weight-nobeyesdad')
    ], style={'display': 'flex', 'justify-content':
'space-around'}),
    html.Div([
        dcc.Graph(id='line-graph-imc-nobeyesdad'),
```

```
    ], style={'display': 'flex', 'justify-content':
'space-around'})
])


# O transporte usado tem uma forte relação com a idade do
indivíduo.
@callback(
    Output('line-graph-age-mtrans', 'figure'),
    Input('line-graph-age-mtrans', 'id')  # O callback precisa de
um Input, mesmo que não seja utilizado
)
def update_line_graph_age_mtrans(value):
    fig = px.line(df_count, x='Age', y='count', color='MTRANS',
                  title='Quantidade de Indivíduos por Idade e Modo
de Transporte',
                  labels={'Age': 'Idade', 'count': 'Quantidade de
Indivíduos', 'MTRANS': 'Modo de Transporte'})
    return fig


# O imc do indivíduo tem uma forte relação com o seu peso.
@callback(
    Output('scatter-graph-weight-imc', 'figure'),
    Input('scatter-graph-weight-imc', 'id')  # O callback precisa
de um Input, mesmo que não seja utilizado
)
def update_scatter_graph_weight_imc(value):
    fig = px.scatter(df, x='Weight', y='imc', title='Relação entre
Peso e IMC',
                     labels={'Weight': 'Peso', 'imc': 'IMC'})
    return fig
```

```
# A altura indivíduo tem uma relação moderada com o peso dele.

# Ajustando o callback para o gráfico de linha
@app.callback(
    Output('bar-graph-height-weight', 'figure'),
    Input('bar-graph-height-weight', 'id')  # O callback precisa de um Input, mesmo que não seja utilizado
)
def update_bar_graph_height_weight(value):

    fig = px.bar(df, x='Weight', y='Height', title='Relação entre Altura e Peso',
                 labels={'Weight': 'Peso', 'Height': 'Altura'})
    return fig


# O peso do indivíduo tem uma relação moderada com o Nível de obesidade.

# Ajustando o callback para o gráfico de linha
@app.callback(
    Output('bar-graph-weight-nobeyesdad', 'figure'),
    Input('bar-graph-weight-nobeyesdad', 'id')  # O callback precisa de um Input, mesmo que não seja utilizado
)
def update_bar_graph_weight_nobeyesdad(value):

    fig = px.bar(df, x='NObeyesdad', y='Weight', title='Relação entre Peso e o Nível de Obesidade',
                 labels={'Weight': 'Peso', 'NObeyesdad':
```

```
'Nível de Obesidade'})
    return fig

# O imc do indivíduo tem uma relação muito forte com o nível de obesidade.
# Ajustando o callback para o gráfico de linha
@app.callback(
    Output('line-graph-imc-nobeyesdad', 'figure'),
    Input('line-graph-imc-nobeyesdad', 'id')  # O callback precisa de um Input, mesmo que não seja utilizado
)
def update_line_graph_imc_nobeyesdad(value):
    fig = px.line(df_count_imc_nob, x='imc', y='count', color='NObeyesdad',
                  title='Quantidade de Indivíduos por IMC e Nível de Obesidade',
                  labels={'count': 'Quantidade de Indivíduos', 'imc': 'IMC', 'NObeyesdad': 'Nível de Obesidade'})
    return fig


if __name__ == '__main__':
    app.run(debug=True)
```

- **Dashboard**



Conclusão

A análise exploratória do conjunto de dados de níveis de obesidade revelou importantes relações entre variáveis como idade, peso, altura, IMC e hábitos alimentares. A visualização interativa fornecida pelo Dash oferece uma maneira eficiente de explorar essas correlações, permitindo uma melhor compreensão dos fatores que influenciam a obesidade. Com base nesses insights, futuras intervenções podem ser direcionadas de forma mais eficaz para combater a obesidade e promover a saúde pública.