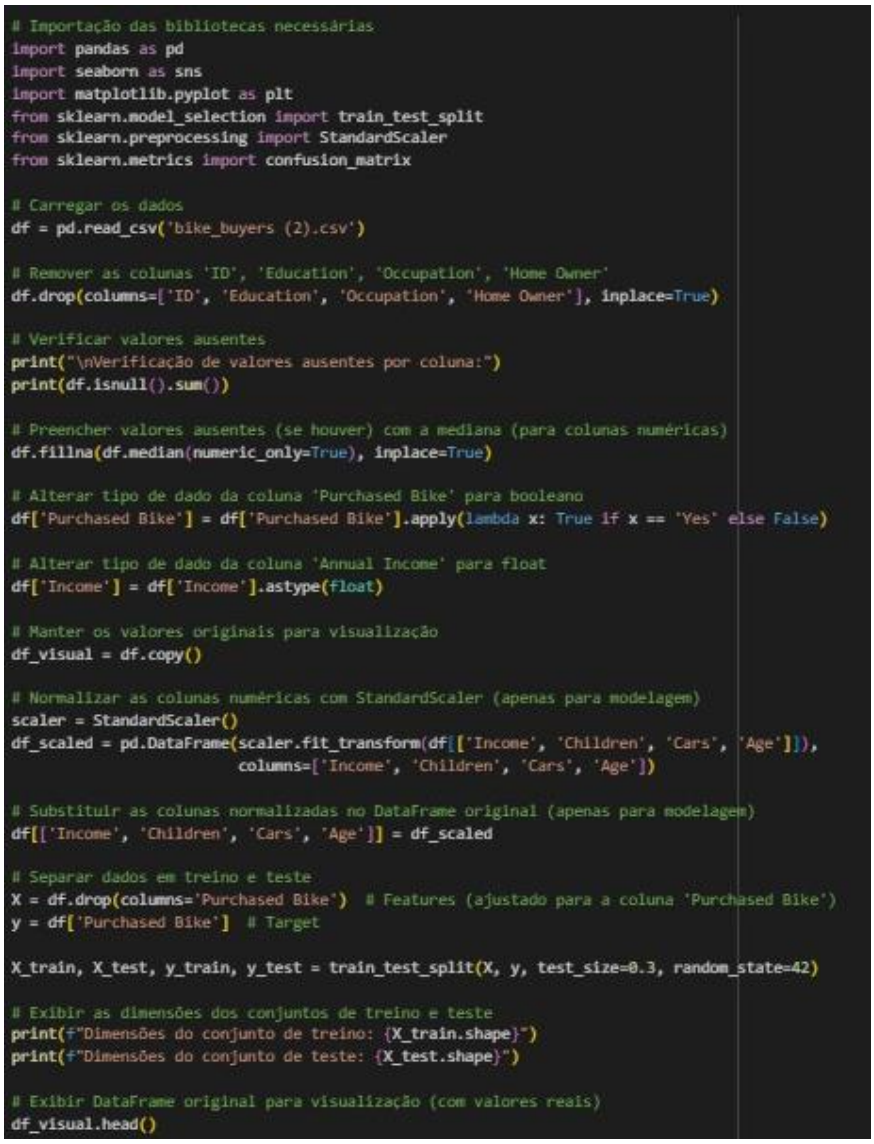
Aprendizado de Máquina, Machine Learning e Ciência de Dados

Entrega 1: Exploração de Dados e Pré-processamento

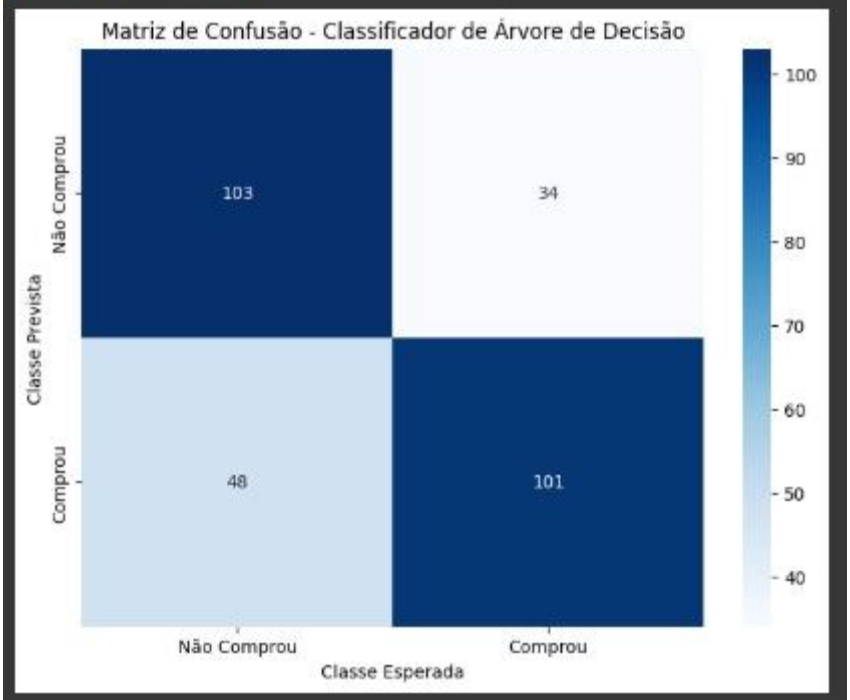


```python
# Importação das bibliotecas necessárias
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix

# Carregar os dados
df = pd.read_csv('bike_buyers (2).csv')

# Remover as colunas 'ID', 'Education', 'Occupation', 'Home Owner'
df.drop(columns=['ID', 'Education', 'Occupation', 'Home Owner'], inplace=True)

# Verificar valores ausentes
print("\nVerificação de valores ausentes por coluna:")
print(df.isnull().sum())

# Preencher valores ausentes (se houver) com a mediana (para colunas numéricas)
df.fillna(df.median(numeric_only=True), inplace=True)

# Alterar tipo de dado da coluna 'Purchased Bike' para booleano
df['Purchased Bike'] = df['Purchased Bike'].apply(lambda x: True if x == 'Yes' else False)

# Alterar tipo de dado da coluna 'Annual Income' para float
df['Income'] = df['Income'].astype(float)

# Manter os valores originais para visualização
df_visual = df.copy()

# Normalizar as colunas numéricas com StandardScaler (apenas para modelagem)
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df[['Income', 'Children', 'Cars', 'Age']]),
                         columns=['Income', 'Children', 'Cars', 'Age'])

# Substituir as colunas normalizadas no DataFrame original (apenas para modelagem)
df[['Income', 'Children', 'Cars', 'Age']] = df_scaled

# Separar dados em treino e teste
X = df.drop(columns='Purchased Bike')  # Features (ajustado para a coluna 'Purchased Bike')
y = df['Purchased Bike']  # Target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Exibir as dimensões dos conjuntos de treino e teste
print(f"Dimensões do conjunto de treino: {X_train.shape}")
print(f"Dimensões do conjunto de teste: {X_test.shape}")

# Exibir DataFrame original para visualização (com valores reais)
df_visual.head()
```

```python
# ////////////////////////////////////////////////////////////////////////////////


import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Carregar o arquivo CSV
df = pd.read_csv('bike_buyers (2).csv')

# Converter a coluna 'Purchased Bike' para valores binários (1 para "Yes" e 0 para "No")
df['Purchased Bike'] = df['Purchased Bike'].map({'Yes': 1, 'No': 0})

# Remover colunas irrelevantes (como ID e name) e lidar com valores ausentes
df = df.drop(['ID', 'name'], axis=1)
df = df.dropna()

# Codificar variáveis categóricas
df = pd.get_dummies(df, drop_first=True)

# Separar variáveis de entrada (X) e variável alvo (y)
X = df.drop('Purchased Bike', axis=1)
y = df['Purchased Bike']

# Dividir os dados em conjunto de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Treinar um classificador de Árvore de Decisão
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Fazer previsões no conjunto de teste
y_pred = clf.predict(X_test)

# Gerar a matriz de confusão
conf_matrix = confusion_matrix(y_test, y_pred)

print("||================================================================||")

# Visualizar a matriz de confusão com mais estilo
print("||🔍 **Análise de Classificação - Modelo: Árvore de Decisão** 🔍  ||")
print("||================================================================||")
print("\n📊 **Matriz de Confusão:**\n")
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
            xticklabels=['Não Comprou', 'Comprou'],
            yticklabels=['Não Comprou', 'Comprou'])
plt.xlabel("Classe Esperada")
plt.ylabel("Classe Prevista")
plt.title("Matriz de Confusão - Classificador de Árvore de Decisão")
plt.show()

# Exibir o relatório de classificação
print("\n📈 **Relatório de Classificação:**\n")
print(classification_report(y_test, y_pred))
```

Análise de Classificação - Modelo: Árvore de Decisão

Matriz de Confusão:



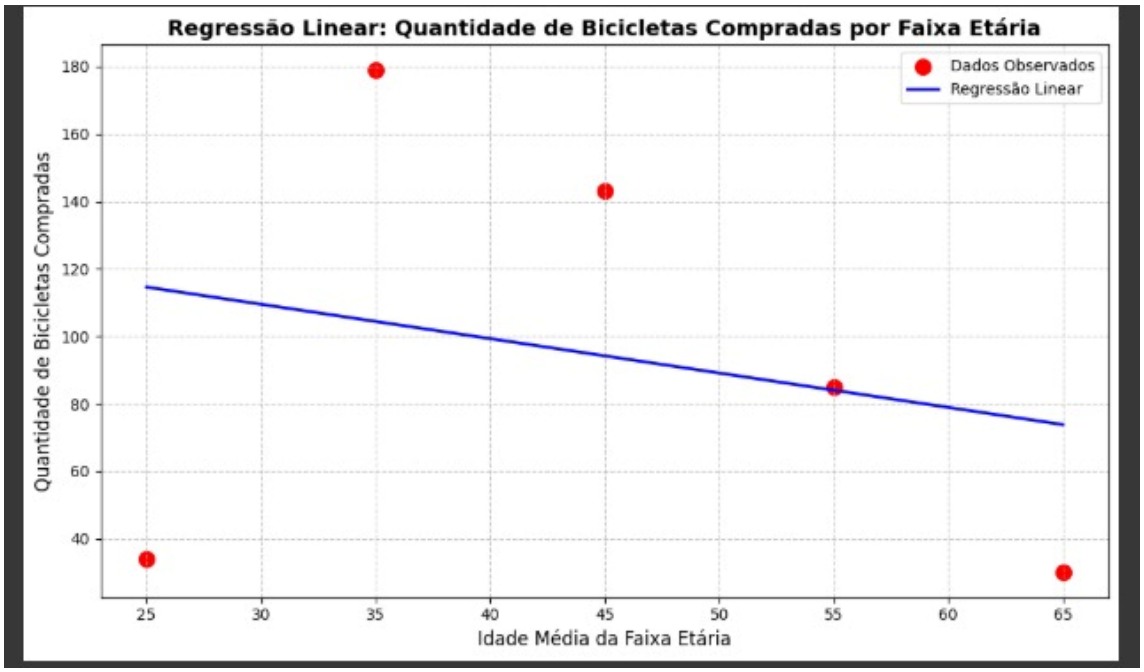
Relatório de Classificação:

	precision	recall	f1-score	support
0	0.68	0.75	0.72	137
1	0.75	0.68	0.71	149
accuracy			0.71	286
macro avg	0.72	0.71	0.71	286
weighted avg	0.72	0.71	0.71	286

Entrega 2: Implementação de Modelos de Aprendizado de Máquina de Regressão Linear

REGRESSÃO LINEAR

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Carregar o arquivo CSV
df = pd.read_csv('bike_buyers (2).csv')

# Definir faixas etárias e rótulos
bins = [20, 30, 40, 50, 60, 70]
labels = ['20-30', '30-40', '40-50', '50-60', '60-70']
df['Age Range'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)

# Contar a quantidade de bicicletas compradas por faixa etária
bike_counts = df[df['Purchased Bike'] == 'Yes'].groupby('Age Range').size()

# Calcular os pontos médios das faixas etárias e os valores ajustados
x_trein = np.array([(bins[i] + bins[i+1]) / 2 for i in range(len(bins) - 1)])
y_trein = np.array(bike_counts.reindex(labels, fill_value=0))

# Regressão linear
N = len(x_trein)
sum_x = np.sum(x_trein)
sum_y = np.sum(y_trein)
sum_xy = np.sum(x_trein * y_trein)
sum_x2 = np.sum(x_trein ** 2)

# Cálculo do peso (w) e viés (b)
w = (N * sum_xy - sum_x * sum_y) / (N * sum_x2 - sum_x ** 2)
b = (sum_y - w * sum_x) / N

# Previsão da linha de regressão
f_wb = w * x_trein + b

# Plotando o gráfico de regressão linear com os dados ajustados
plt.figure(figsize=(10, 6))

# Plotando os dados observados (pontos vermelhos)
plt.scatter(x_trein, y_trein, color='red', marker='o', s=100, label='Dados Observados')

# Plotando a linha de regressão
plt.plot(x_trein, f_wb, color='blue', label='Regressão Linear', linewidth=2)

# Ajustando rótulos e título
plt.ylabel('Quantidade de Bicicletas Compradas', fontsize=12)
plt.xlabel('Idade Média da Faixa Etária', fontsize=12)
plt.title('Regressão Linear: Quantidade de Bicicletas Compradas por Faixa Etária', fontsize=14, fontweight='bold')
plt.legend()

# Melhorando a grade do gráfico
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()

# Exibir o gráfico
plt.show()
```


Regressão Linear: Quantidade de Bicicletas Compradas por Faixa Etária
Dados Observados
Regressão Linear
Quantidade de Bicicletas Compradas
Idade Média da Faixa Etária
180
160
140
120
100
80
60
40
25
30
35
40
45
50
55
60
65

Entrega 3: Implementação de Modelos de Aprendizado de Máquina de Classificação

CLASSIFICAÇÃO POR MATPLOT

```python
# MATPLOT DAS COMPRAS DAS BICICLETAS
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns

# Converte 'Purchased Bike' para valores numéricos, se necessário
df['Purchased Bike'] = df['Purchased Bike'].map({'Yes': 1, 'No': 0}).fillna(df['Purchased Bike'])

# Calcula a quantidade de compradores e não compradores
purchased_count = df['Purchased Bike'].sum()
not_purchased_count = len(df) - purchased_count

# Define os dados para o gráfico de pizza
numbers = [purchased_count, not_purchased_count]
labels = ["Bicicletas compradas", "Bicicletas não compradas"]
colors = ["#2ca02c", "#d62728"]  # Usando cores mais agradáveis
explode = (0.05, 0)  # Destacando a fatia "Purchased Bike"

# Cria o gráfico de pizza com configurações aprimoradas
plt.figure(figsize=(6, 6))  # Ajustando o tamanho da figura
plt.pie(numbers, labels=labels, autopct='%1.1f%%', colors=colors,
        explode=explode, shadow=True, startangle=90, textprops={'fontsize': 12})

# Adiciona um título informativo
plt.title("Proporção de Compradores de Bicicletas", fontsize=14)

# Centraliza o gráfico
plt.axis('equal')

# Exibe o gráfico
plt.show()
```