

Curso:	Ciência de Dados e Inteligência Artificial	Aluno:	Juliano França da Mata
Modular:	5º	RA:	23200190
Disciplina:	Projeto Integrador V A	Data Entrega:	13/04/2025

Introdução

A linguagem de programação Python, lançada em 1991, destaca-se por permitir o desenvolvimento de soluções com menos linhas de código em comparação a outras linguagens. Sua simplicidade, legibilidade e vasta comunidade contribuíram para sua ampla adoção em diversas áreas da tecnologia. Atualmente, o Python está integrado em praticamente todas as novas tecnologias, sendo utilizado em aplicações web, mobile, data science, machine learning, blockchain, entre outras.

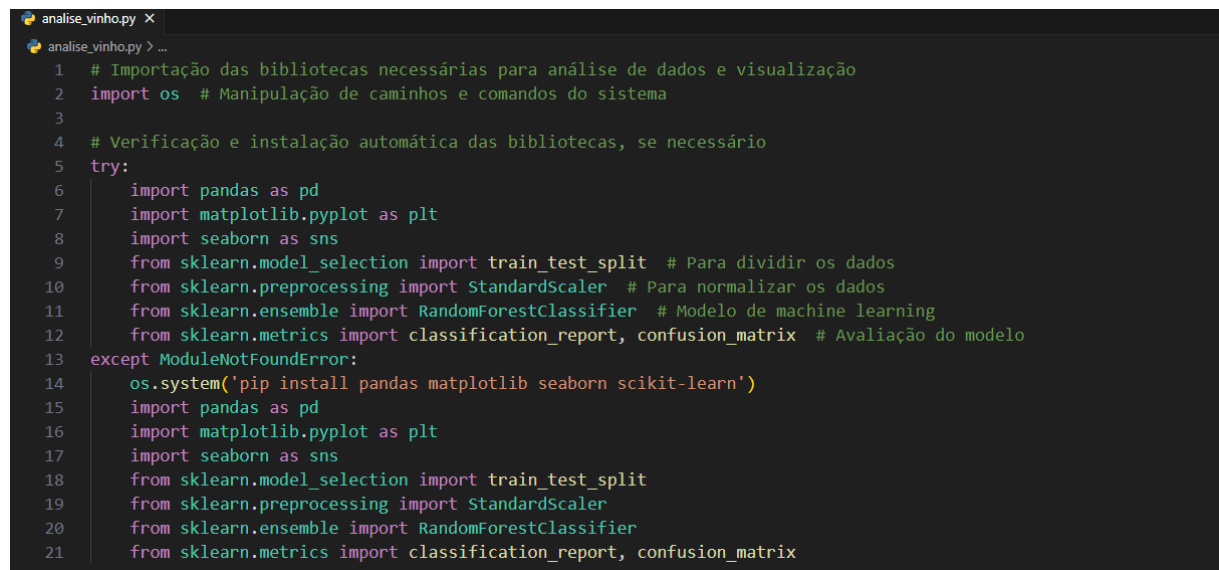
Desde 2009, o Python tornou-se a linguagem padrão do curso de Ciência da Computação do Massachusetts Institute of Technology (MIT), o que reforça sua importância no meio acadêmico e científico. Além disso, foi eleito "Linguagem do Ano" pelo índice TIOBE em diferentes ocasiões: 2007, 2010, 2018 e 2020, demonstrando sua constante evolução e relevância no cenário da programação contemporânea (TIOBE, 2020).

O presente projeto integrador tem como base um tutorial publicado no site Data Flair, que aborda o pré-processamento e visualização de dados com Python. A primeira etapa do estudo contempla técnicas como normalização, padronização, transformação e binarização de dados,

utilizando os pacotes Pandas e Scikit-learn. Já a segunda etapa do tutorial explora a visualização de dados por meio de histogramas e gráficos de densidade.

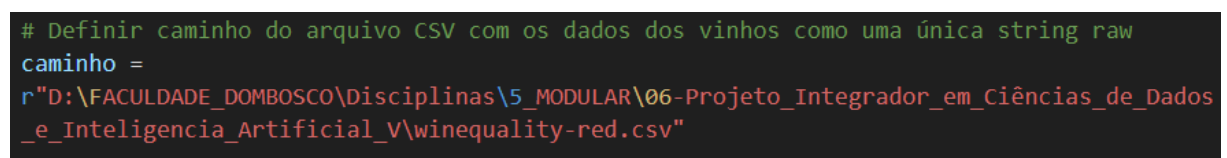
A base de dados utilizada refere-se à análise de vinhos portugueses e está disponível no repositório UCI Machine Learning Repository, na versão winequality-red.csv, a qual também se encontra nos arquivos da disciplina na plataforma Canvas. Como parte da proposta do projeto, será elaborado um relatório técnico no formato ABNT, contendo os programas desenvolvidos e os gráficos gerados a partir das operações realizadas sobre os dados.

Desenvolvimento



```
analise_vinho.py X
analise_vinho.py > ...
1 # Importação das bibliotecas necessárias para análise de dados e visualização
2 import os # Manipulação de caminhos e comandos do sistema
3
4 # Verificação e instalação automática das bibliotecas, se necessário
5 try:
6     import pandas as pd
7     import matplotlib.pyplot as plt
8     import seaborn as sns
9     from sklearn.model_selection import train_test_split # Para dividir os dados
10    from sklearn.preprocessing import StandardScaler # Para normalizar os dados
11    from sklearn.ensemble import RandomForestClassifier # Modelo de machine learning
12    from sklearn.metrics import classification_report, confusion_matrix # Avaliação do modelo
13 except ModuleNotFoundError:
14     os.system('pip install pandas matplotlib seaborn scikit-learn')
15     import pandas as pd
16     import matplotlib.pyplot as plt
17     import seaborn as sns
18     from sklearn.model_selection import train_test_split
19     from sklearn.preprocessing import StandardScaler
20     from sklearn.ensemble import RandomForestClassifier
21     from sklearn.metrics import classification_report, confusion_matrix
22
```

Fig.01



```
# Definir caminho do arquivo CSV com os dados dos vinhos como uma única string raw
caminho =
r"D:\FACULDADE_DOMBOSCO\Disciplinas\5_MODULAR\06-Projeto_Integrador_em_Ciências_de_Dados_e_Inteligencia_Artificial_V\winequality-red.csv"
```

Fig.02

```
# 1. Carregar os dados do arquivo CSV (separador ';' usado nesse dataset)
df = pd.read_csv(caminho, sep=';')
df.columns = df.columns.str.strip() # Remover espaços em branco dos nomes das colunas
```

Fig.03

```
# 2. Visualizar as primeiras linhas do dataset para ter uma ideia da estrutura dos dados
print("Visualizando as primeiras linhas do dataset:")
print(df.head())
```

Fig.04

Visualizando as primeiras linhas do dataset:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

Fig.05

```
# 3. Exibir informações gerais sobre os dados: tipo de dado, valores não nulos etc.
print("\nInformações gerais do dataset:")
print(df.info())
```

Fig.06

```
Informações gerais do dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   fixed acidity          1599 non-null   float64
1   volatile acidity       1599 non-null   float64
2   citric acid            1599 non-null   float64
3   residual sugar         1599 non-null   float64
4   chlorides              1599 non-null   float64
5   free sulfur dioxide    1599 non-null   float64
6   total sulfur dioxide   1599 non-null   float64
7   density                1599 non-null   float64
8   pH                    1599 non-null   float64
9   sulphates              1599 non-null   float64
10  alcohol                1599 non-null   float64
11  quality                1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
None
```

Fig.07

```
# 4. Estatísticas descritivas básicas: média, desvio padrão, mínimo, máximo etc.
print("\nEstatísticas descritivas:")
print(df.describe())
```

Fig.08

Estatísticas descritivas:													
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.636023	
std	1.741896	0.179860	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.807569	
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000	
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000	
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000	
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000	
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000	

Fig.09

```
# 5. Verificar se há valores ausentes em alguma coluna
total_nulos = df.isnull().sum()
print("\nValores ausentes por coluna:")
print(total_nulos)
if total_nulos.sum() == 0:
    print("\nNão há valores ausentes no dataset.")
```

Fig.10

```
Valores ausentes por coluna:
fixed acidity      0
volatile acidity   0
citric acid        0
residual sugar     0
chlorides          0
free sulfur dioxide 0
total sulfur dioxide 0
density            0
pH                0
sulphates          0
alcohol            0
quality            0
dtype: int64

Não há valores ausentes no dataset.
```

Fig.11

```
# 6. Visualizar a distribuição da variável alvo 'quality' (qualidade dos vinhos)
plt.figure(figsize=(8, 5))
sns.countplot(x='quality', data=df, hue='quality', palette='viridis', legend=False)
plt.title("Distribuição da Qualidade do Vinho")
plt.xlabel("Qualidade")
plt.ylabel("Contagem")
plt.show()
```

Fig.12

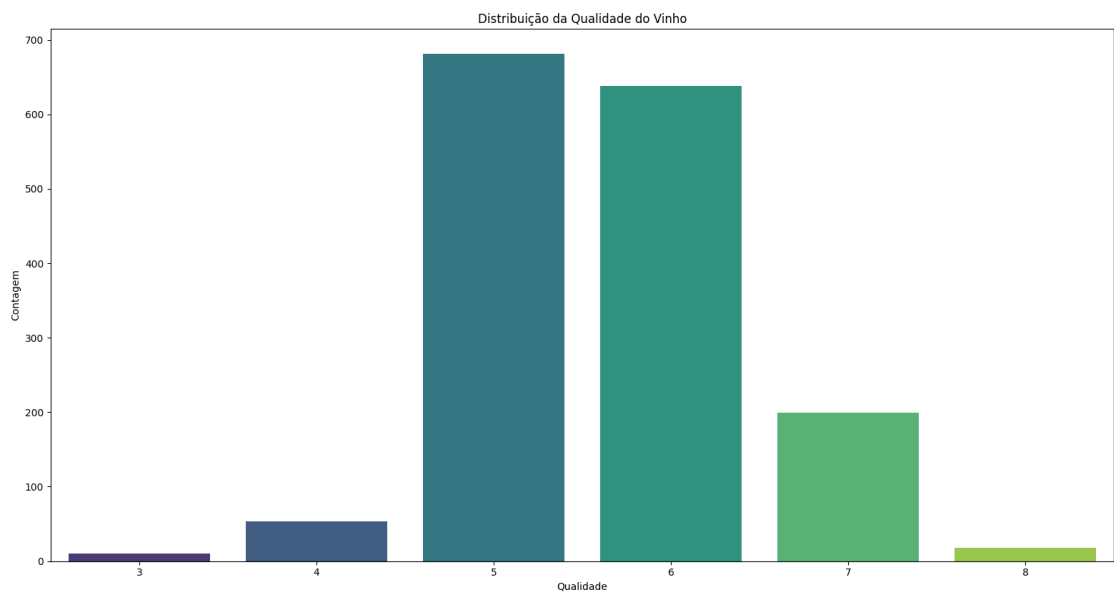


Fig.13

```
# 7. Matriz de correlação: identificar relações entre variáveis numéricas
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title("Matriz de Correlação das Variáveis")
plt.show()
```

Fig.14

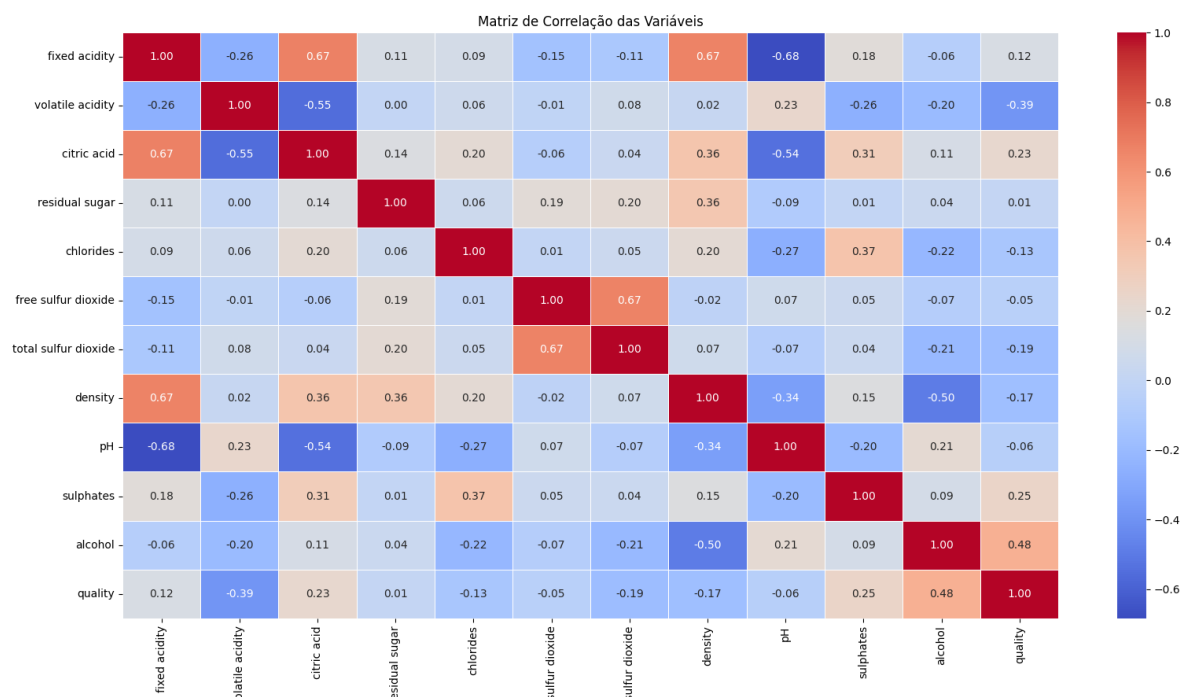


Fig.15

```
# 8. Distribuição da acidez fixa para entender sua densidade
plt.figure(figsize=(8, 5))
df['fixed acidity'].hist(bins=30, color='skyblue', edgecolor='black')
plt.title("Distribuição da Acidez Fixa")
plt.xlabel("Acidez Fixa")
plt.ylabel("Frequência")
plt.show()
```

Fig.16

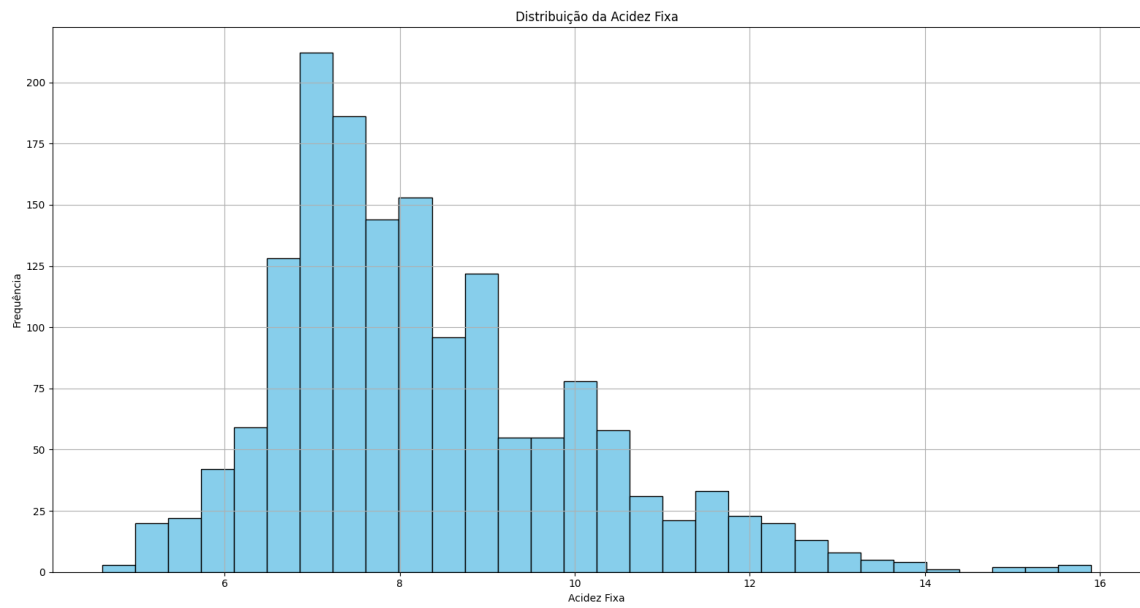


Fig.17

```
# 9. Boxplots para detectar outliers em variáveis importantes
plt.figure(figsize=(12, 6))
sns.boxplot(data=df[['alcohol', 'volatile acidity', 'residual sugar', 'pH']], palette='Set2')
plt.title("Boxplot de Variáveis para Identificação de Outliers")
plt.show()
```

Fig.18

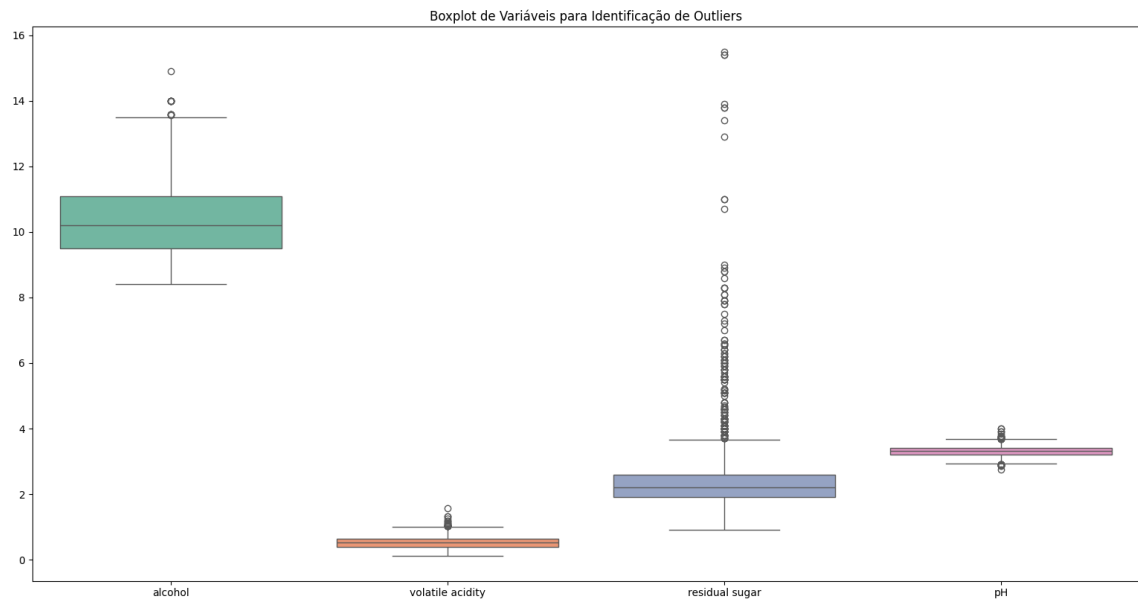


Fig.19

```
# 10. Relação entre teor alcoólico e qualidade do vinho
plt.figure(figsize=(10, 6))
sns.boxplot(x='quality', y='alcohol', data=df, hue='quality', palette='magma', legend=False)
plt.title("Relação entre Teor Alcoólico e Qualidade")
plt.xlabel("Qualidade")
plt.ylabel("Teor Alcoólico")
plt.show()
```

Fig.20

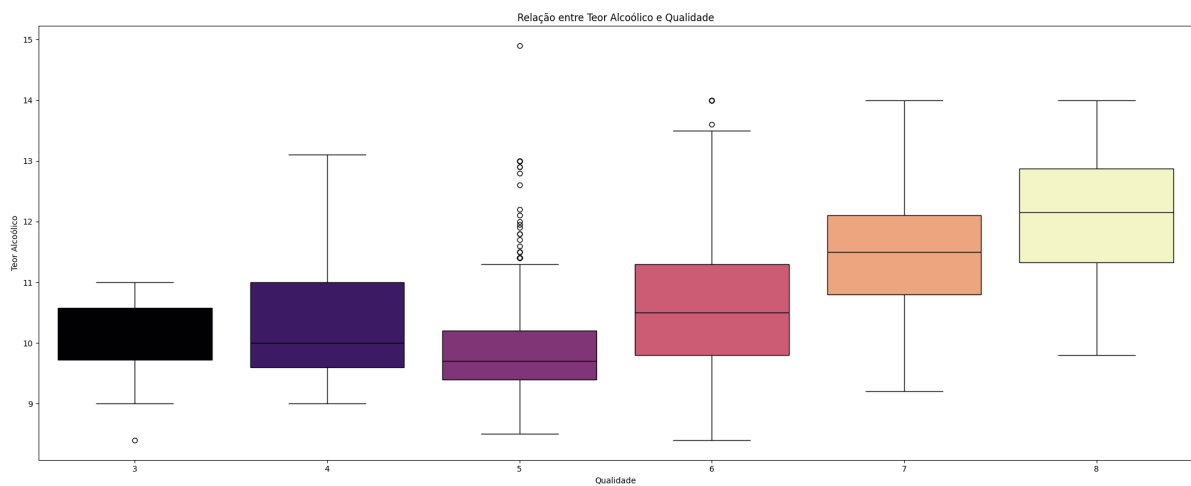


Fig.21

```
# 11. Agrupar a variável 'quality' em categorias: baixa (0), média (1), alta (2)
def categorizar_qualidade(valor):
    if valor <= 4:
        return 0 # baixa
    elif valor <= 6:
        return 1 # média
    else:
        return 2 # alta

df['qualidade_cat'] = df['quality'].apply(categorizar_qualidade)

# Visualizar nova distribuição após o agrupamento
plt.figure(figsize=(8, 5))
sns.countplot(x='qualidade_cat', data=df, hue='qualidade_cat', palette='plasma', legend=False)
plt.title("Distribuição Agrupada da Qualidade do Vinho")
plt.xlabel("Categorias de Qualidade (0=Baixa, 1=Média, 2=Alta)")
plt.ylabel("Contagem")
plt.show()
```

Fig.22

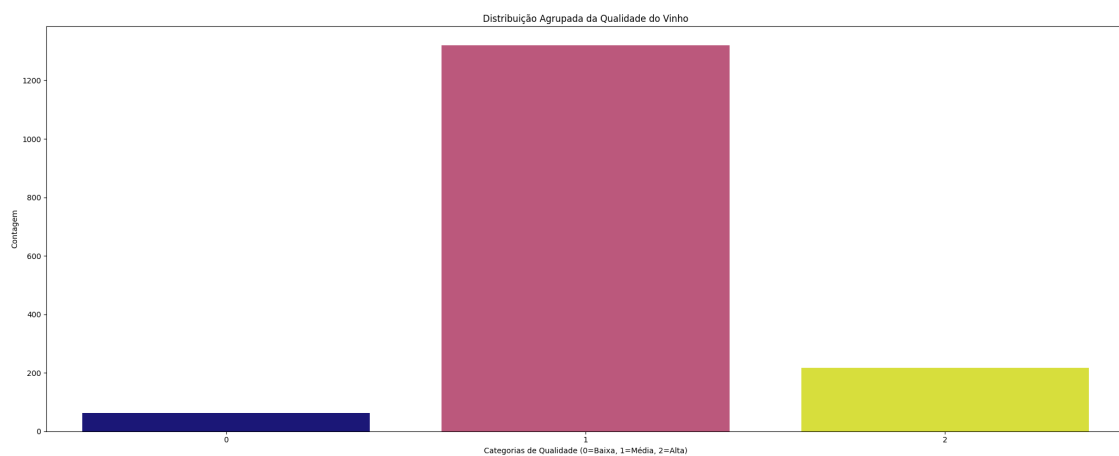


Fig.23


```
# 12. Preparação dos dados para o modelo de machine learning
X = df.drop(['quality', 'qualidade_cat'], axis=1) # Atributos (features)
y = df['qualidade_cat'] # Variável alvo categorizada

# Normalização dos dados
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Dividir em conjunto de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Fig.24

```
# 13. Treinar um modelo de classificação (Random Forest)
modelo = RandomForestClassifier(n_estimators=100, random_state=42)
modelo.fit(X_train, y_train)
```

Fig.25

```
# 14. Avaliar o modelo
y_pred = modelo.predict(X_test)
print("\nMatriz de Confusão:")
print(confusion_matrix(y_test, y_pred))
print("\nRelatório de Classificação:")
print(classification_report(y_test, y_pred, zero_division=0))
```

Fig.26

```
Matriz de Confusão:
[[ 0 11  0]
 [ 0 250 12]
 [ 0 21 26]]
```

Fig.27

```
# Isso significa:
# - Dos 11 vinhos realmente da Classe 0, todos foram classificados
  erroneamente como Classe 1.
# - Dos 262 vinhos da Classe 1, 250 foram corretamente classificados e
  12 foram confundidos com Classe 2.
# - Dos 47 vinhos da Classe 2, 26 foram corretamente classificados, mas
  21 foram confundidos com Classe 1.

"""Embora a Classe 0 não tenha sido corretamente classificada pelo
modelo, esse comportamento é aceitável dentro do escopo educacional do
projeto. Futuramente, técnicas de balanceamento de dados ou ajuste de
hiperparâmetros poderiam ser exploradas para melhorar a performance."""
```

Fig.28

Relatório de Classificação:				
	precision	recall	f1-score	support
0	0.00	0.00	0.00	11
1	0.89	0.95	0.92	262
2	0.68	0.55	0.61	47
accuracy			0.86	320
macro avg	0.52	0.50	0.51	320
weighted avg	0.83	0.86	0.84	320

Fig.29

```
# Relatório de Classificação:
# Mostra o desempenho do modelo em cada classe (0 = Ruim, 1 = Regular,
# 2 = Boa)
# Métricas:
# - precision: acertos entre os que foram previstos como aquela classe
# - recall: acertos entre os que realmente pertencem àquela classe
# - f1-score: equilíbrio entre precision e recall
# - support: total real de exemplos daquela classe

# Observações:
# - O modelo teve ótimo desempenho na classe "Regular" (classe 1)
# - Teve dificuldades nas classes "Ruim" (0) e "Boa" (2), possivelmente
#   por poucos exemplos (desequilíbrio de classes)
# - A acurácia geral foi de 86%, o que é um bom resultado para um
#   projeto educacional
# - A média ponderada das métricas também indica desempenho consistente
#   no geral

""" Por ser um exercício inicial, não faremos ajustes finos no modelo
neste momento"""
```

Fig.30

Conclusão:

- A análise inicial forneceu uma visão geral do conjunto de dados, incluindo a estrutura e estatísticas básicas.
- A matriz de correlação identificou variáveis com maior influência na qualidade do vinho.

- A distribuição da acidez e da qualidade revelou padrões importantes.
- Boxplots ajudaram a detectar outliers que podem impactar modelos futuros.
- A relação entre teor alcoólico e qualidade mostrou uma tendência clara de maior qualidade com mais álcool.
- Também foi aplicado um modelo de Random Forest para prever a qualidade do vinho com base nas variáveis do dataset.
- Neste caso, a qualidade foi agrupada em três categorias (baixa, média e alta) para facilitar a predição.
- As métricas de avaliação ajudam a entender a performance e precisão do modelo com essa nova abordagem.

Próximas etapas recomendadas:

- Otimização de hiperparâmetros do modelo (com GridSearchCV, por exemplo).
- Teste com outros algoritmos como SVM, KNN, ou redes neurais
- Análise de importância das variáveis para compreender quais características mais influenciam na qualidade.

Referências

- [Wine Quality Dataset – UCI](#)
- [Documentação do Scikit-learn](#)
- [Guia oficial do Pandas](#)
- [Seaborn: Visualização estatística em Python](#)
- [Faculdade Dom Bosco](#)
- [Github – JulianoMata](#)