Atividade 2 - Aprendizado de Máquina

AUTHOR
Rennan Dalla Guimarães

Introdução

Caso esteja queira acessar o código fonte ou a versão em HTML, .PY ou .QMD, acesse o repositório da atividade e os arquivos para essa atividade serão os nomeados como "Aula2_rennanguimaraes".

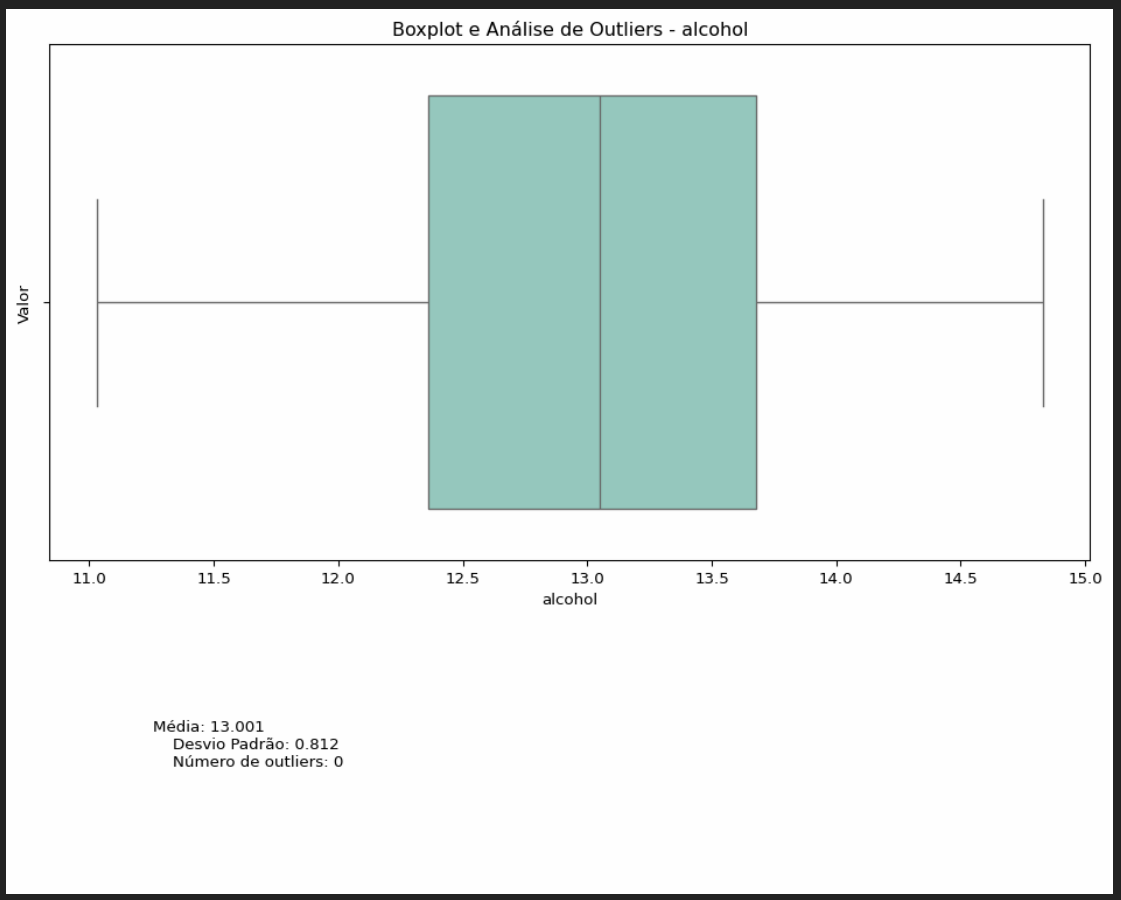
Imports e Dataset

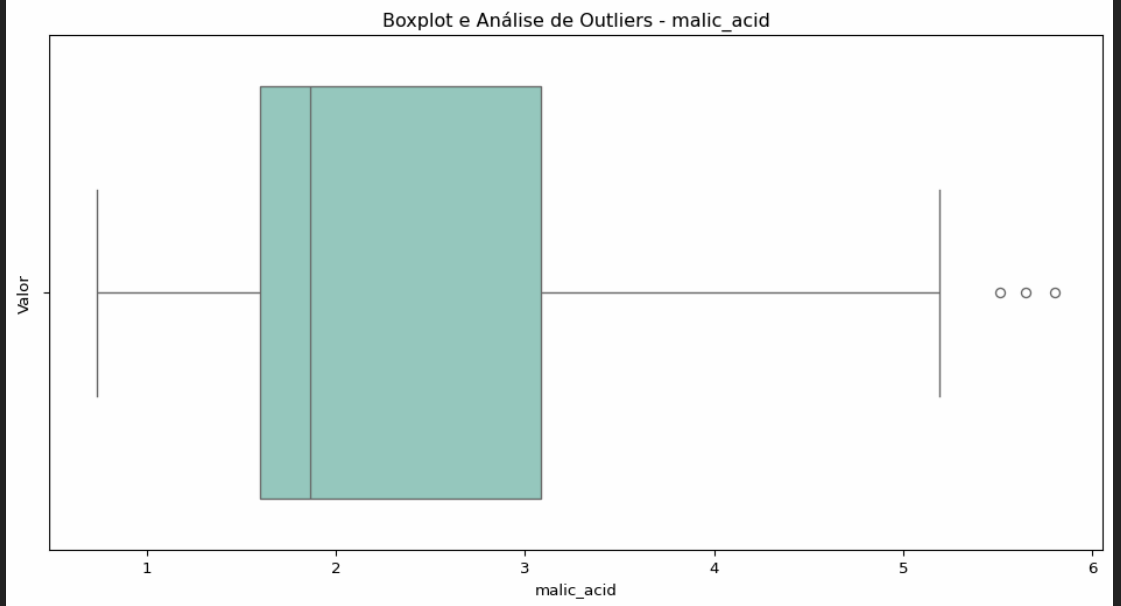
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.datasets import load_wine
from sklearn.decomposition import PCA
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
import warnings

# Carregar dataset e silenciar warnings
wine_dataset = load_wine()
wine_features = pd.DataFrame(wine_dataset.data,
  columns=wine_dataset.feature_names)
wine_target = pd.Series(wine_dataset.target, name="class")
warnings.filterwarnings("ignore")
```

1a

```
for feature_name in wine_features.columns:
    # Limites para outliers (IQR)
    Q1 = wine_features[feature_name].quantile(0.25)
    Q3 = wine_features[feature_name].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = wine_features[
        (wine_features[feature_name] < lower_bound)
        | (wine_features[feature_name] > upper_bound)
    ][feature_name]

    # Figura com dois subplots
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8),
      height_ratios=[2, 1])

    # Boxplot
    sns.boxplot(x=feature_name, data=wine_features,
    palette="Set3", ax=ax1, legend=False)
```

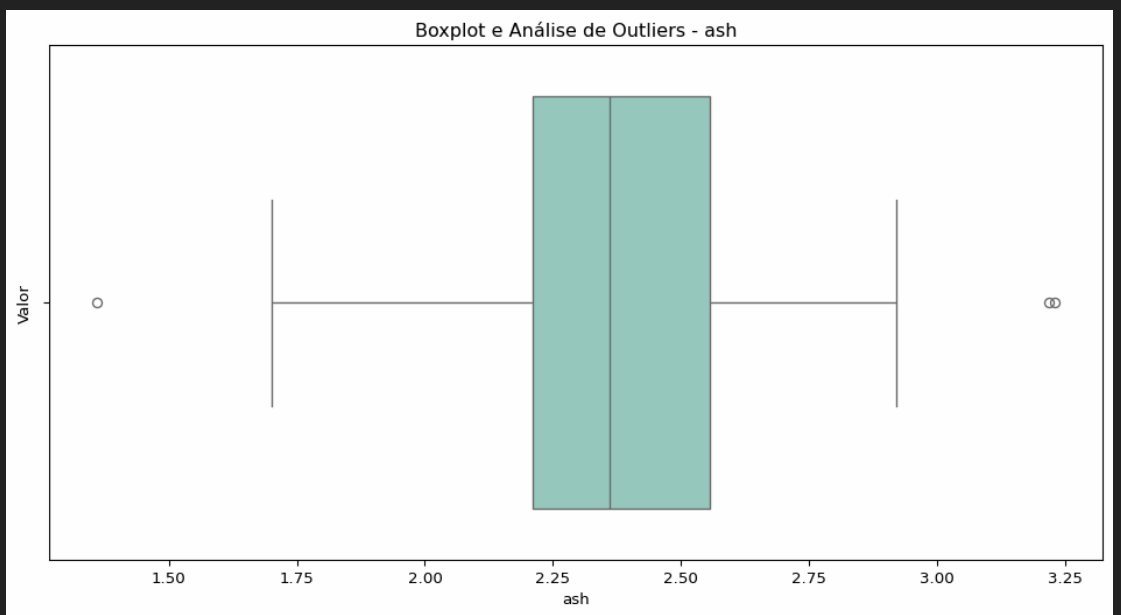
```
ax1.set_title(f"Boxplot e Análise de Outliers - {feature_n
ax1.set_ylabel("Valor")

# Estatísticas
ax2.axis("off")
stats_text = f"""Média: {wine_features[feature_name].mean(
Desvio Padrão: {wine_features[feature_name].std():.3f}
Número de outliers: {len(outliers)}
"""
if len(outliers) > 0:
    stats_text += f"\nValores dos outliers:\n{outliers.val

ax2.text(0.1, 0.5, stats_text, fontsize=10, va="center")

plt.tight_layout()
plt.show()
```





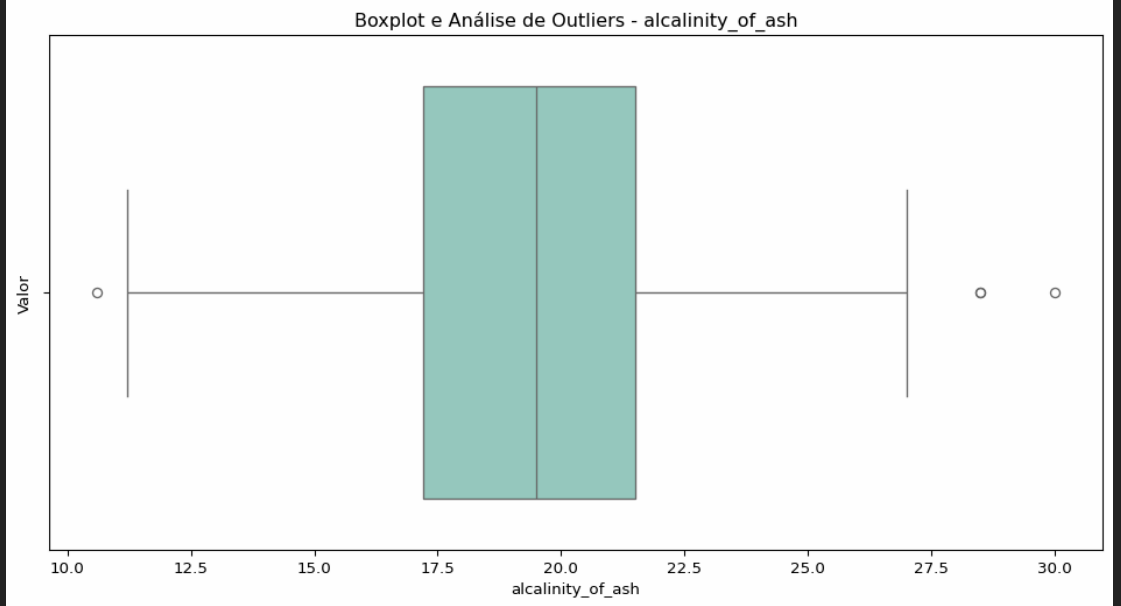
Média: 2.336
Desvio Padrão: 1.117
Número de outliers: 3

Valores dos outliers:
[5.8 5.51 5.65]



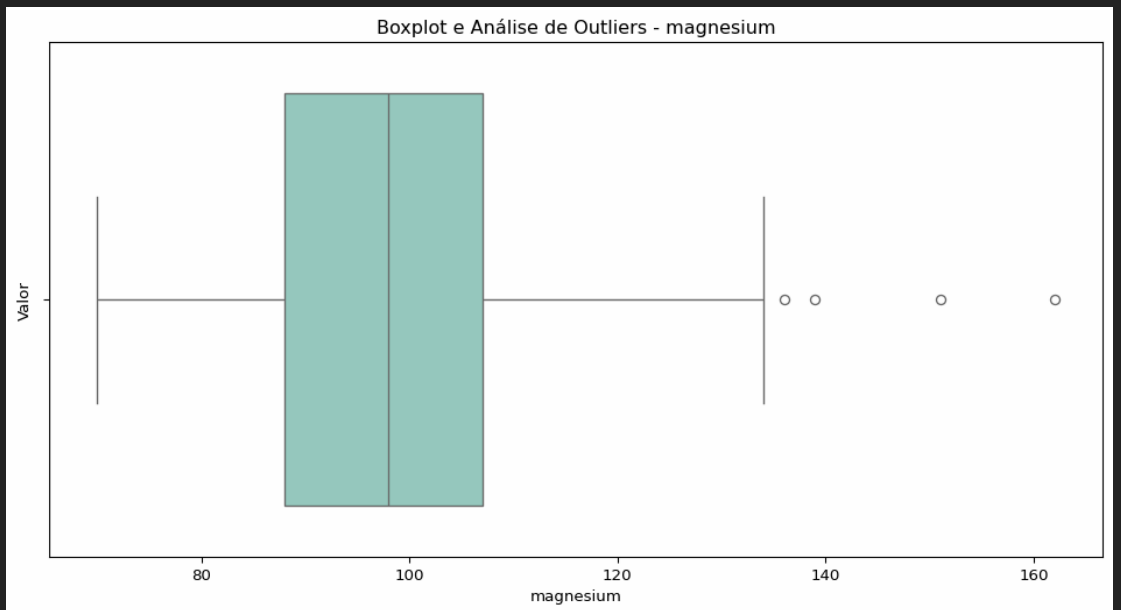
Média: 2.367
Desvio Padrão: 0.274
Número de outliers: 3

Valores dos outliers:
[3.22 1.36 3.23]



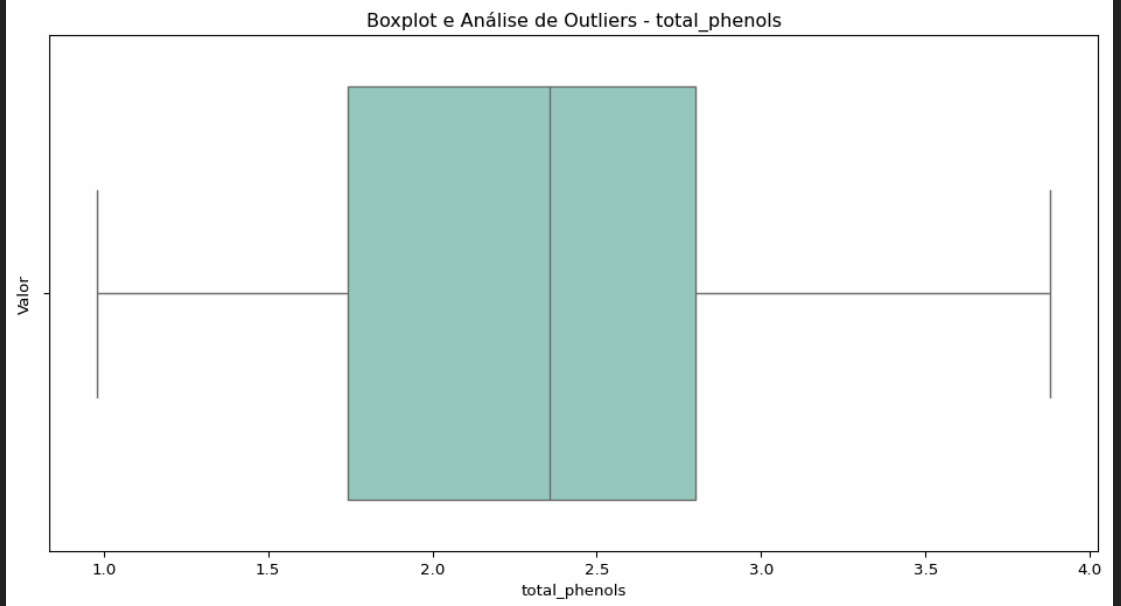
Média: 19.495
Desvio Padrão: 3.340
Número de outliers: 4

Valores dos outliers:
[10.6 30. 28.5 28.5]

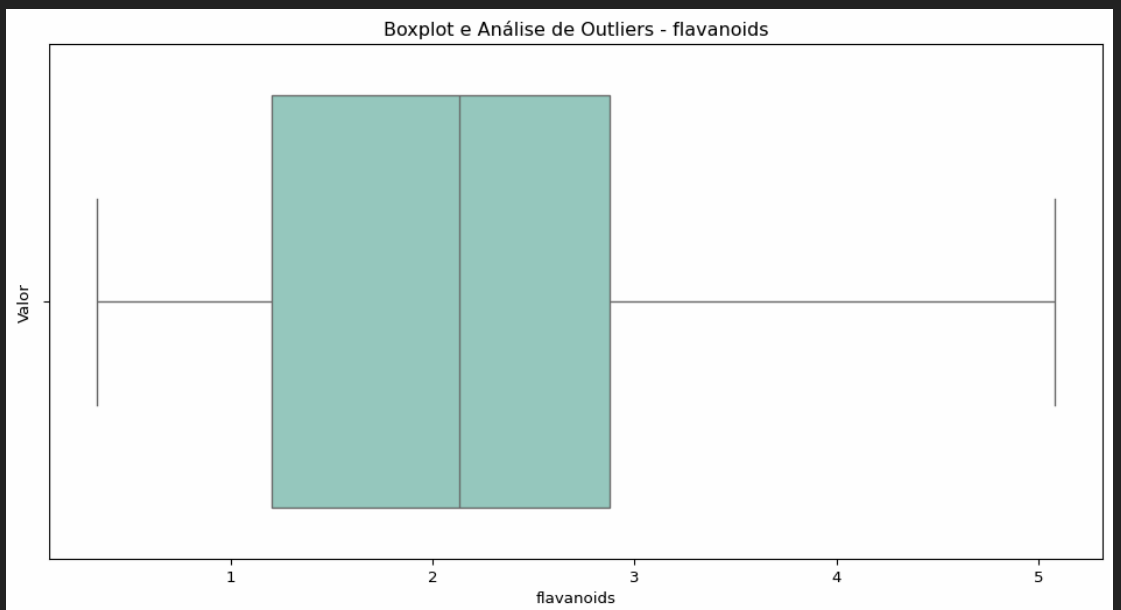


Média: 99.742
Desvio Padrão: 14.282
Número de outliers: 4

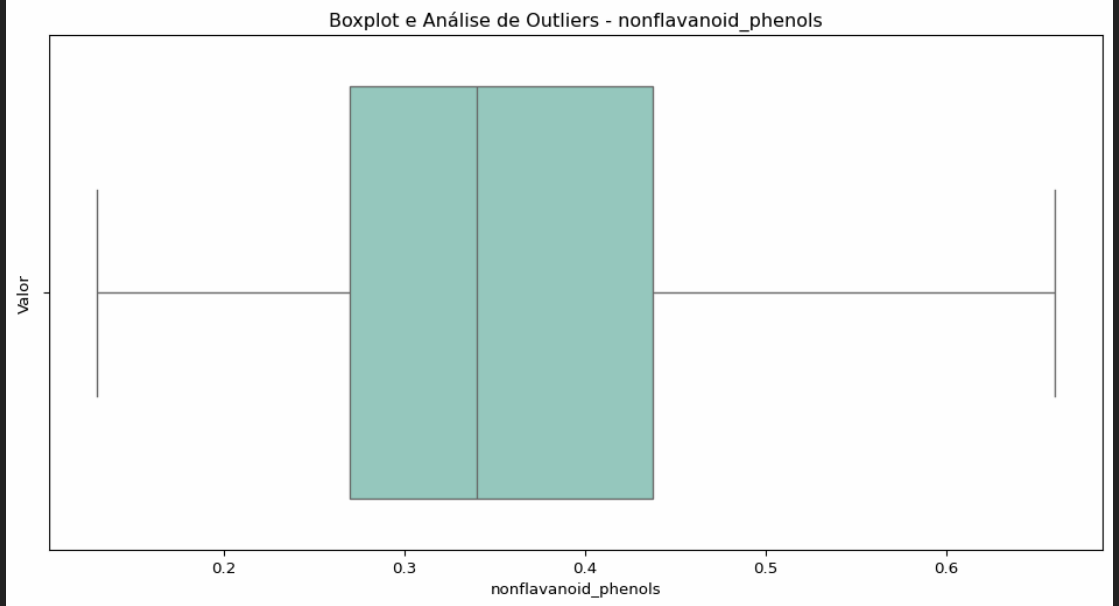
Valores dos outliers:
[151. 139. 136. 162.]



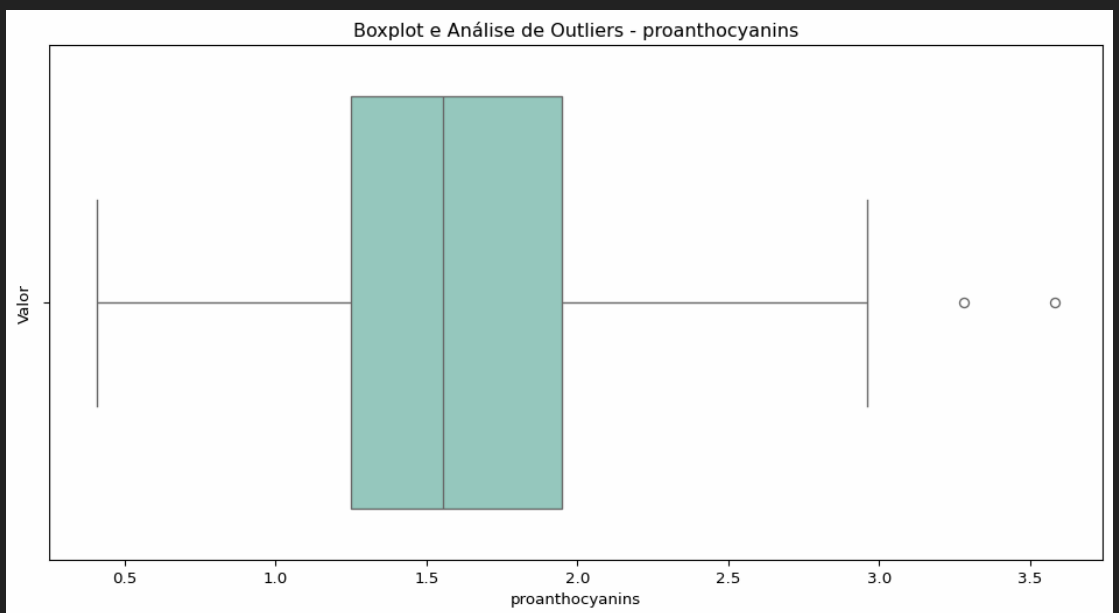
Média: 2.295
Desvio Padrão: 0.626
Número de outliers: 0



Média: 2.029
Desvio Padrão: 0.999
Número de outliers: 0

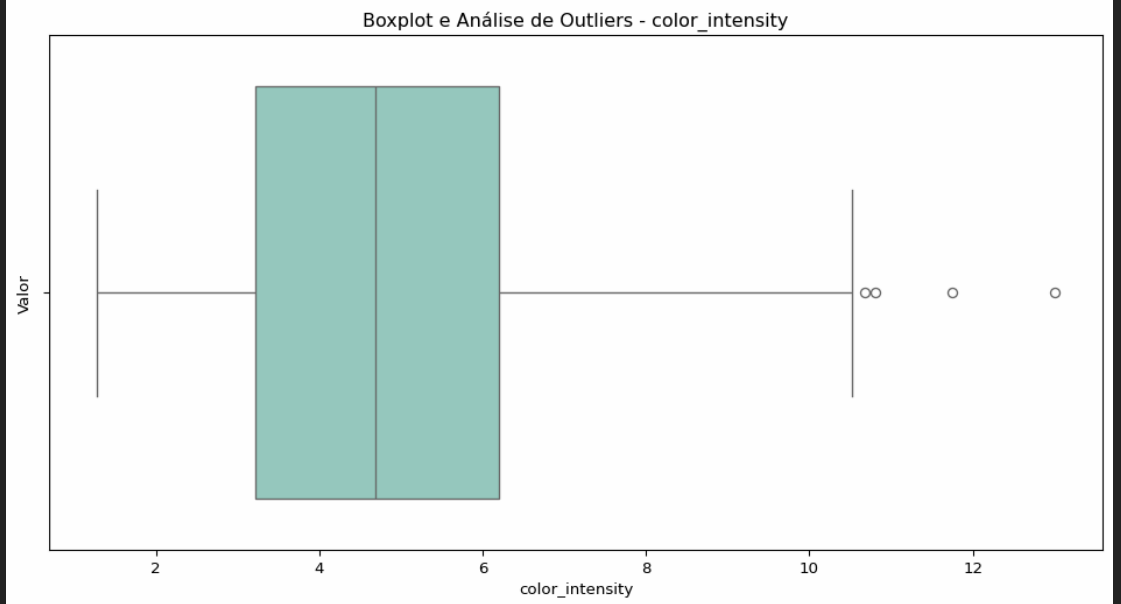


Média: 0.362
Desvio Padrão: 0.124
Número de outliers: 0



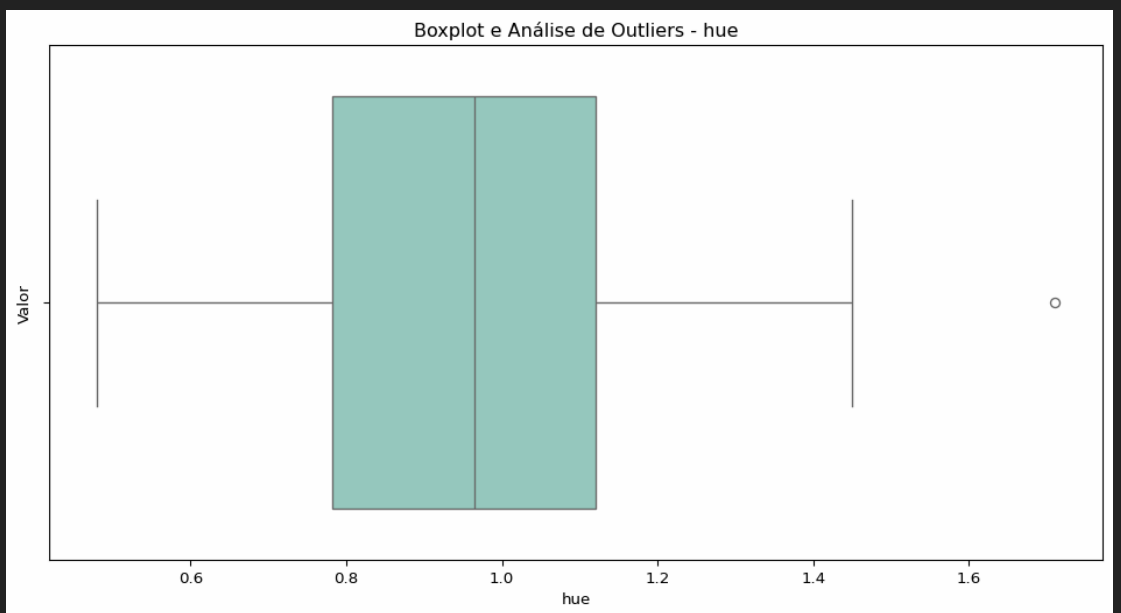
Média: 1.591
Desvio Padrão: 0.572
Número de outliers: 2

Valores dos outliers:
[3.28 3.58]



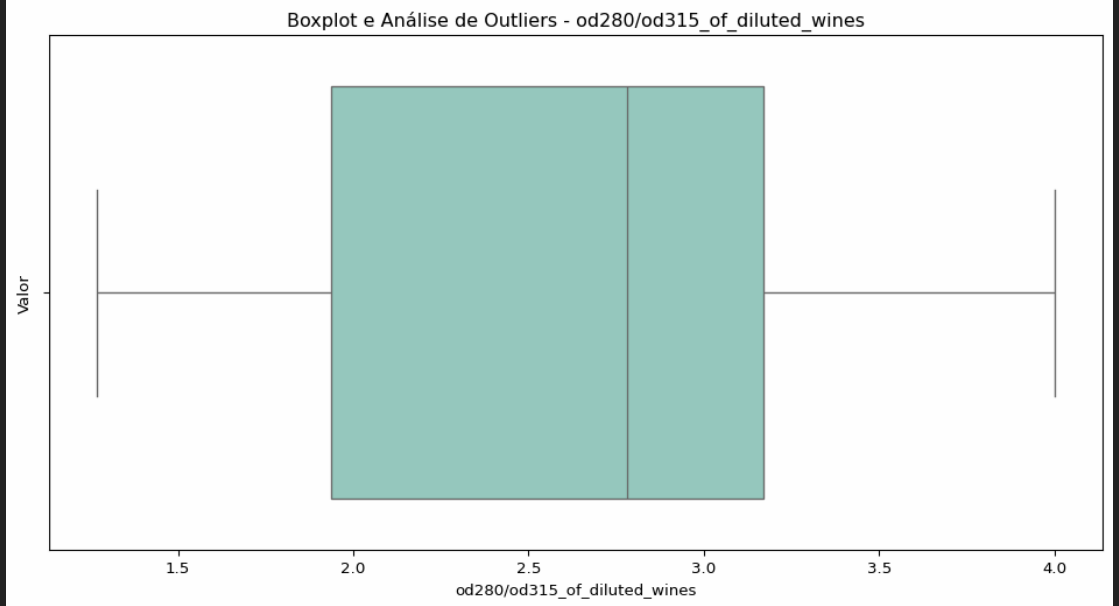
Média: 5.058
Desvio Padrão: 2.318
Número de outliers: 4

Valores dos outliers:
[10.8 13. 11.75 10.68]

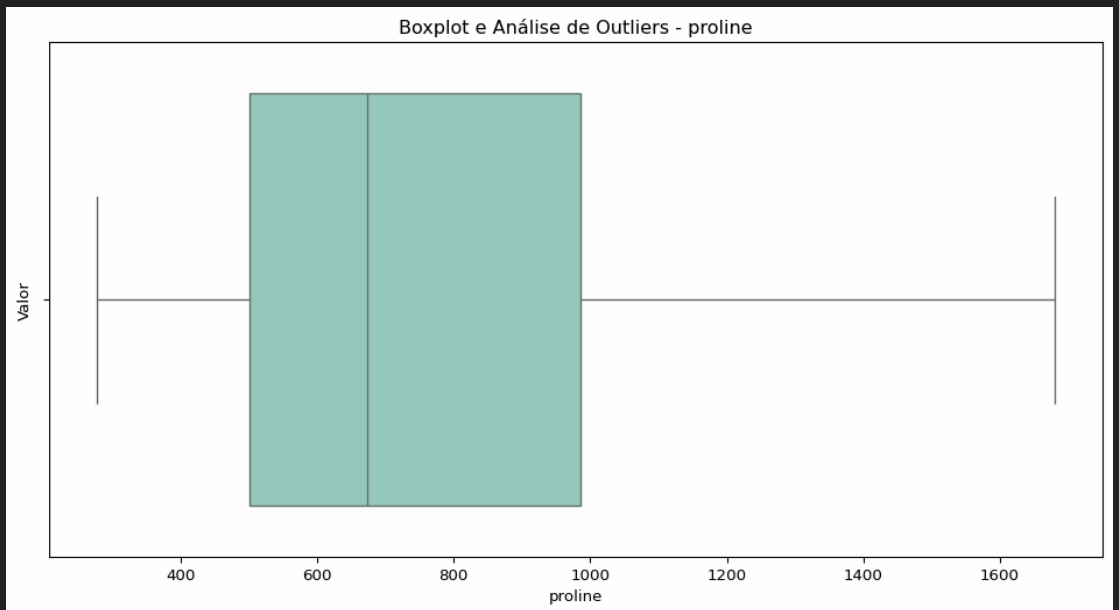


Média: 0.957
Desvio Padrão: 0.229
Número de outliers: 1

Valores dos outliers:
[1.71]



Média: 2.612
Desvio Padrão: 0.710
Número de outliers: 0



Média: 746.893
Desvio Padrão: 314.907
Número de outliers: 0

1b

```
min_max_scaler = MinMaxScaler()
normalized_features = pd.DataFrame(
    min_max_scaler.fit_transform(wine_features),
    columns=wine_dataset.feature_names
)
```

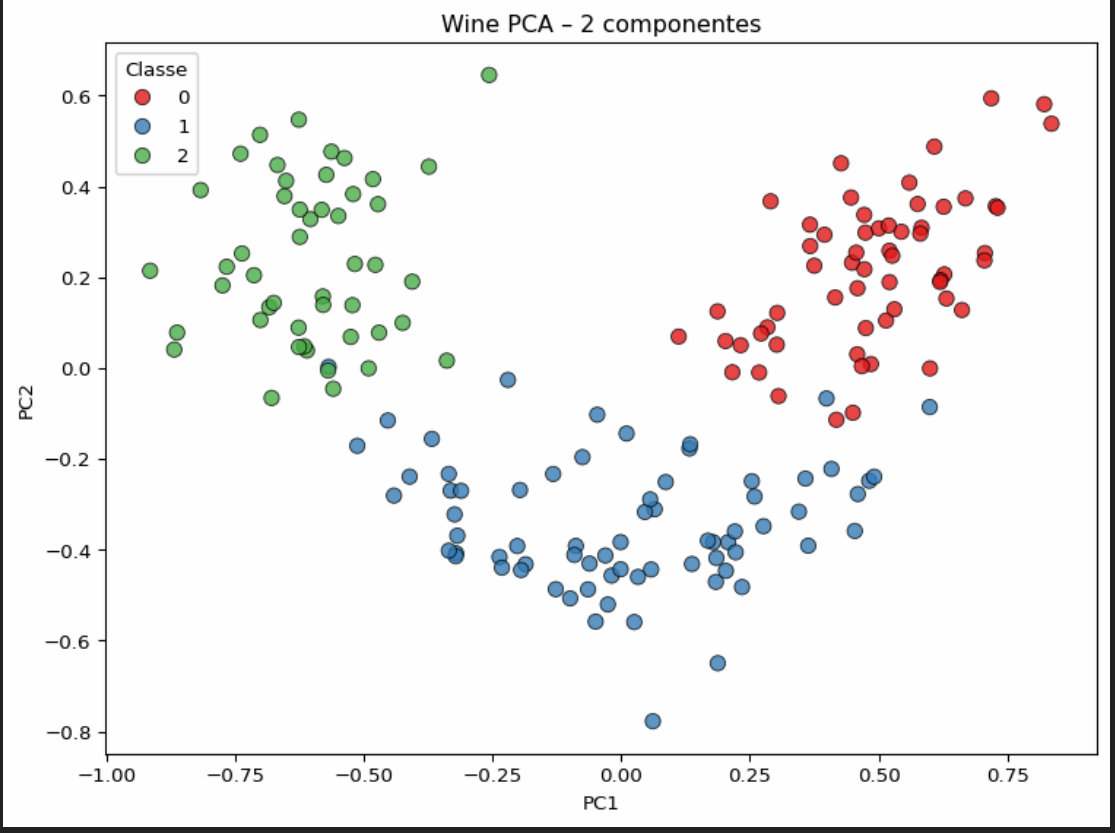
```
        # Mostrar primeiras linhas
        normalized_features.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nor
0	0.842105	0.191700	0.572193	0.257732	0.619565	0.627586	0.573840	0.2
1	0.571053	0.205534	0.417112	0.030928	0.326087	0.575862	0.510549	0.2
2	0.560526	0.320158	0.700535	0.412371	0.336957	0.627586	0.611814	0.3
3	0.878947	0.239130	0.609626	0.319588	0.467391	0.989655	0.664557	0.2
4	0.581579	0.365613	0.807487	0.536082	0.521739	0.627586	0.495781	0.4

1c

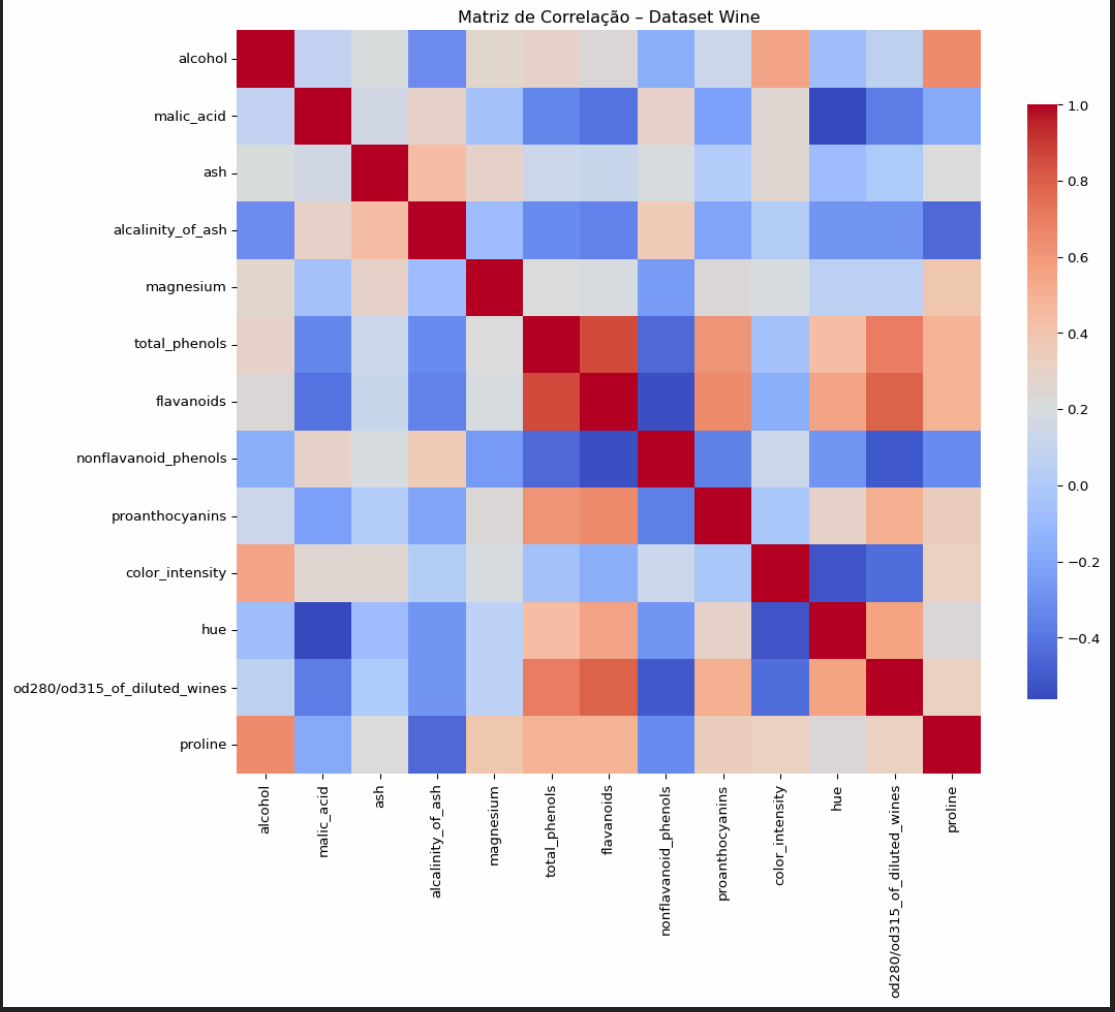
```
        pca_transformer = PCA(n_components=2, random_state=42)
        pca_features = pca_transformer.fit_transform(normalized_featur

        print(f"""Variância explicada pelos 2 componentes:
          {pca_transformer.explained_variance_ratio_.sum():.2%}""")

        plt.figure(figsize=(8, 6))
        color_palette = sns.color_palette("Set1",
        n_colors=len(np.unique(wine_target)))
        sns.scatterplot(
            x=pca_features[:, 0],
            y=pca_features[:, 1],
            hue=wine_target,
            palette=color_palette,
            s=60,
            alpha=0.8,
            edgecolor="k",
        )
        plt.xlabel("PC1")
        plt.ylabel("PC2")
        plt.title("Wine PCA — 2 componentes")
        plt.legend(title="Classe")
        plt.tight_layout()
        plt.show()
```

```
Variância explicada pelos 2 componentes:
  59.72%
```



1d

```
correlation_matrix = wine_features.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(
    correlation_matrix,
    annot=False,
    cmap="coolwarm",
    square=True,
    cbar_kws={"shrink": 0.8},
)
plt.title("Matriz de Correlação – Dataset Wine")
plt.tight_layout()
plt.show()

# Pares de features com correlação >= 0.75
correlation_pairs = (
    correlation_matrix.where(
        np.triu(np.ones(correlation_matrix.shape), k=1).astype
    )
    .stack()
    .reset_index()
)
correlation_pairs.columns = ["feature_1", "feature_2", "correl
high_correlation_pairs = correlation_pairs.loc[
    correlation_pairs["correlation"].abs() >= 0.75
]
high_correlation_pairs
```



	feature_1	feature_2	correlation
50	total_phenols	flavanoids	0.864564
61	flavanoids	od280/od315_of_diluted_wines	0.787194

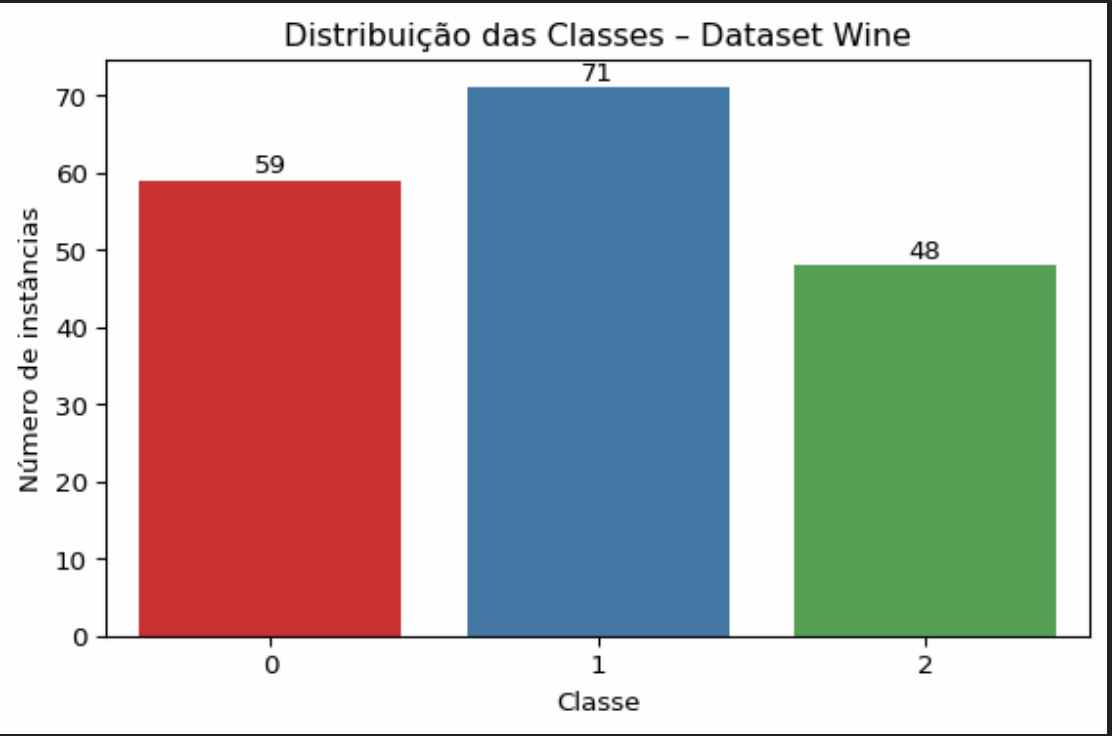
1e

```
        class_distribution = wine_target.value_counts().sort_index()
        class_distribution
```

```
class
0    59
1    71
2    48
Name: count, dtype: int64
```

```
        plt.figure(figsize=(6, 4))
        sns.barplot(x=class_distribution.index, y=class_distribution.v
          palette=color_palette)
        plt.xlabel("Classe")
        plt.ylabel("Número de instâncias")
        plt.title("Distribuição das Classes – Dataset Wine")
        for idx, val in enumerate(class_distribution.values):
            plt.text(idx, val + 1, str(val), ha="center")
        plt.tight_layout()
        plt.show()
```

```
class_percentage = class_distribution.values / len(wine_target
class_percentage
```




```
array([33.14606742, 39.88764045, 26.96629213])
```

2

```
features_with_missing = wine_features.copy()

# Remover 5% dos valores aleatoriamente
rng = np.random.default_rng(42)
missing_mask = rng.random(features_with_missing.shape) < 0.05
features_with_missing = features_with_missing.mask(missing_mas

missing_percentage = features_with_missing.isna().mean().mean(
print(f"""Percentual total de valores ausentes
  introduzidos: {missing_percentage:.2f}%""")

# Imputar usando mediana
median_imputer = SimpleImputer(strategy="median")
imputed_features = pd.DataFrame(
    median_imputer.fit_transform(features_with_missing),
    columns=wine_dataset.feature_names,
)
imputed_features.head()
```

```
Percentual total de valores ausentes
  introduzidos: 4.80%
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanc
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26
2	13.16	1.83	2.67	18.6	101.0	2.80	3.24	0.30

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanc
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39

Referências

- Slides
- Código de referência passado em aula
- Chat gpt para melhorar eficiencia do código, como por exemplo os loops e retirar dúvidas de entendimento da documentação
- Links consultados:
  - https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.boxplot.html
  - https://seaborn.pydata.org/generated/seaborn.boxplot.html
  - https://numpy.org/doc/2.1/reference/generated/numpy.quantile.html
  - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.quantile.html
  - https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
  - https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
  - https://seaborn.pydata.org/generated/seaborn.scatterplot.html
  - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html
  - https://seaborn.pydata.org/generated/seaborn.heatmap.html
  - https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html
  - https://seaborn.pydata.org/generated/seaborn.barplot.html
  - https://numpy.org/doc/2.2/reference/random/generator.html
  - https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html