Escolha uma base de dados. Essa base deve ser multivariada e possuir uma variável objetivo, que pode ser usada na fase de treino (modelos supervisionados) e/ou na validação do modelo (modelos supervisionados e não-supervisionados) OBS: Caso você não possua uma base, utilize a base de dados de vinho brancos. Considere bons vinhos aqueles que obtiveram notas >= 6. Como motivo da escolha (questão 2.a) apenas indique utilizou a base proposta pelo professor. Explique a origem dos dados e o motivo para a escolha. Descreva, também, como os dados foram obtidos. Essa é a fase de COMPREENSÃO DO NEGÓCIO.

RESPONDENDO:

Utilizando vinhos brancos, bons com notas >=6 . Motivo: base proposta pelo professor.

Esta é uma base real, apresentada no artigo: P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

- Link para o artigo : https://www.sciencedirect.com/science/article/abs/pii/S0167923609001377
- Para a base de dados: https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv

Descreva as variáveis do problema e o tipo de cada uma (categórica ou numérica). Mostre a distribuição (usando um histograma) para cada uma delas. Comente sobre a faixa dinâmica de cada uma delas. Essa é a fase de COMPREENSÃO DOS DADOS.

RESPONDENDO:

Compreensão do Negócio: A base de dados de vinhos brancos é uma escolha comum para análise e modelagem na área de Ciência de Dados, devido à sua disponibilidade e variedade de informações relevantes. Esses dados são úteis para avaliar a qualidade dos vinhos brancos e identificar padrões que possam influenciar essa qualidade.

Compreensão dos Dados: As variáveis presentes na base de dados de vinhos brancos incluem características físico-químicas, como acidez, teor alcoólico, pH, dentre outras, além da variável objetivo, que é a qualidade do vinho. A maioria das variáveis é numérica, enquanto a qualidade é uma variável categórica ordinal (mais detalhes na tabela abaixo).

Para obter uma compreensão inicial dos dados, podemos criar histogramas (abaixo) para visualizar a distribuição de cada variável e sua faixa dinâmica. Isso nos ajudará a entender a amplitude e a dispersão dos valores em cada variável, bem como possíveis desequilíbrios nos dados.

Descreva o objetivo do modelo que será criado neste projeto. Agora faça o tratamento de dados necessários para o treinamento: escalonamento, normalização, transformação de variável (ex: aplicar a função log em uma variável com distribuição exponencial), separação entre treino e

teste… Você deve optar pelas tarefas necessárias, descrevendo cada uma delas e justificando o motivo. Essa é a fase PREPARAÇÃO DOS DADOS.

RESPONDENDO:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random
import numpy as np
from sklearn.preprocessing import StandardScaler, PowerTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, StratifiedKFold
from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    roc_curve,
    auc,
    RocCurveDisplay
)
```

```
wines = pd.read_csv('winequalityN.csv')
wines.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   type                  6497 non-null   object 
 1   fixed acidity         6487 non-null   float64
 2   volatile acidity      6489 non-null   float64
 3   citric acid           6494 non-null   float64
 4   residual sugar        6495 non-null   float64
 5   chlorides             6495 non-null   float64
 6   free sulfur dioxide   6497 non-null   float64
 7   total sulfur dioxide  6497 non-null   float64
 8   density               6497 non-null   float64
 9   pH                    6488 non-null   float64
 10  sulphates             6493 non-null   float64
 11  alcohol               6497 non-null   float64
 12  quality               6497 non-null   int64  
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
```

```
removed_rows = (len(wines.dropna()) / len(wines)) - 1
print(f"Remover os valores com NaN diminui {abs(removed_rows) * 100:.2f}% da base original
wines = wines.dropna()
```

```
Remover os valores com NaN diminui 0.52% da base original de vinhos.
```

```
wines.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6463 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   type                  6463 non-null   object 
 1   fixed acidity         6463 non-null   float64
 2   volatile acidity      6463 non-null   float64
 3   citric acid           6463 non-null   float64
 4   residual sugar        6463 non-null   float64
 5   chlorides             6463 non-null   float64
 6   free sulfur dioxide   6463 non-null   float64
 7   total sulfur dioxide  6463 non-null   float64
 8   density               6463 non-null   float64
 9   pH                    6463 non-null   float64
 10  sulphates             6463 non-null   float64
 11  alcohol               6463 non-null   float64
 12  quality               6463 non-null   int64  
dtypes: float64(11), int64(1), object(1)
memory usage: 706.9+ KB
```

```
wines['opinion'] = [0 if quality <6 else 1 for quality in wines['quality']]
wines.head()
```

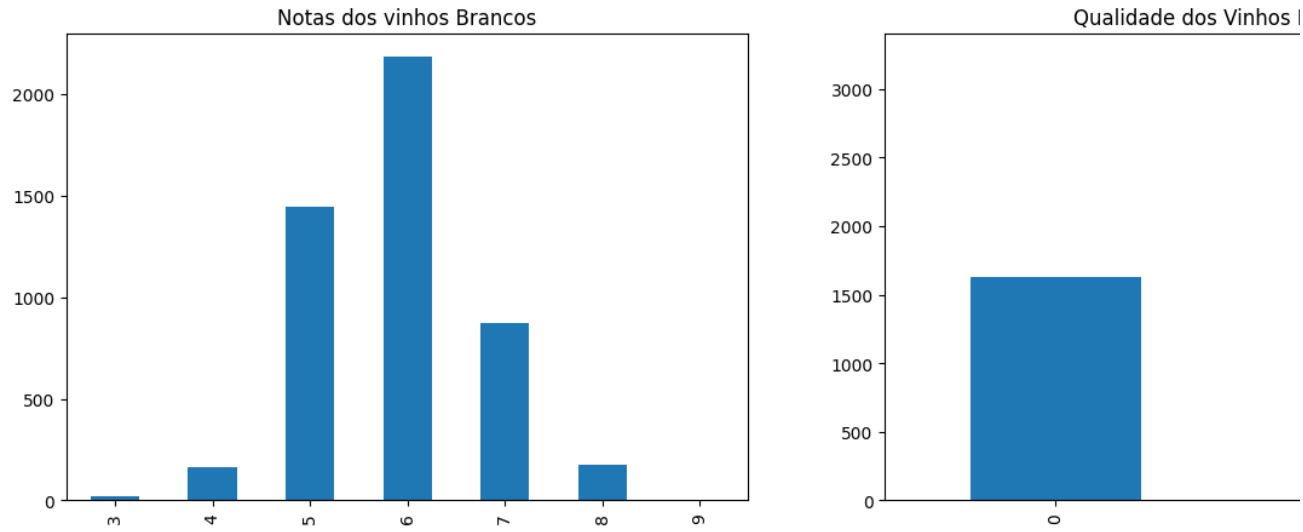
	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3

```
wines.type.value_counts()
```

```
white    4870
red      1593
Name: type, dtype: int64
```

```
# Modelo para vinhos brancos (white wines) - como foi pedido apenas vinhos brancos, retira

white_wines = wines[wines.type == 'white'].reset_index().drop('index', axis=1)

fig, axs = plt.subplots(1, 2, figsize=(16, 5))
ax = plt.subplot(121)
white_wines.quality.value_counts().sort_index(ascending=True).plot.bar()
ax.set_title('Notas dos vinhos Brancos')
ax = plt.subplot(122)
```

```
white_wines.opinion.value_counts().sort_index(ascending=True).plot.bar()
ax.set_title('Qualidade dos Vinhos Brancos');
```




```
# Descrição das variáveis

white_wines.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	
count	4870.000000	4870.000000	4870.000000	4870.000000	4870.000000	4870.000000	48
mean	6.855123	0.278071	0.334199	6.394343	0.045771	35.317146	1
std	0.843444	0.100528	0.120915	5.070853	0.021846	17.012967	
min	3.800000	0.080000	0.000000	0.600000	0.009000	2.000000	
25%	6.300000	0.210000	0.270000	1.700000	0.036000	23.000000	1
50%	6.800000	0.260000	0.320000	5.200000	0.043000	34.000000	1
75%	7.300000	0.320000	0.390000	9.900000	0.050000	46.000000	1
max	14.200000	1.100000	1.660000	65.800000	0.346000	289.000000	4

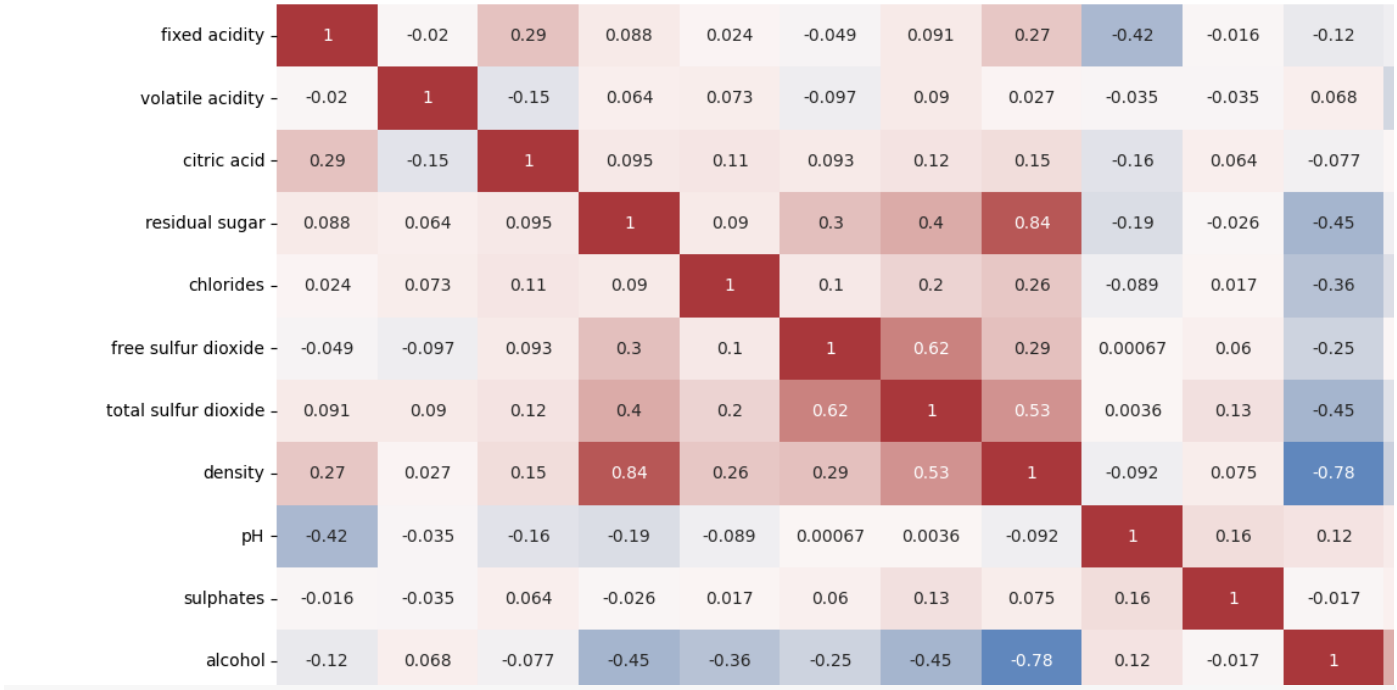
Variável	Tipo	Valor Médio	Desvio Padrão
type	Categórica	-	-
fixed acidity	Contínua	6.85	0.84
volatile acidity	Contínua	0.27	0.10

Variável	Tipo	Valor Médio	Desvio Padrão
citric acid	Contínua	0.33	0.12
residual sugar	Contínua	6.39	5.07
chlorides	Contínua	0.04	0.02
free sulfur dioxide	Contínua	35.3	17.01
total sulfur dioxide	Contínua	138.34	42.49
density	Contínua	0.99	0.0029
pH	Contínua	3.18	0.15
sulphates	Contínua	0.48	0.11
quality	Categórica	-	-
opinion	Categórica	-	-

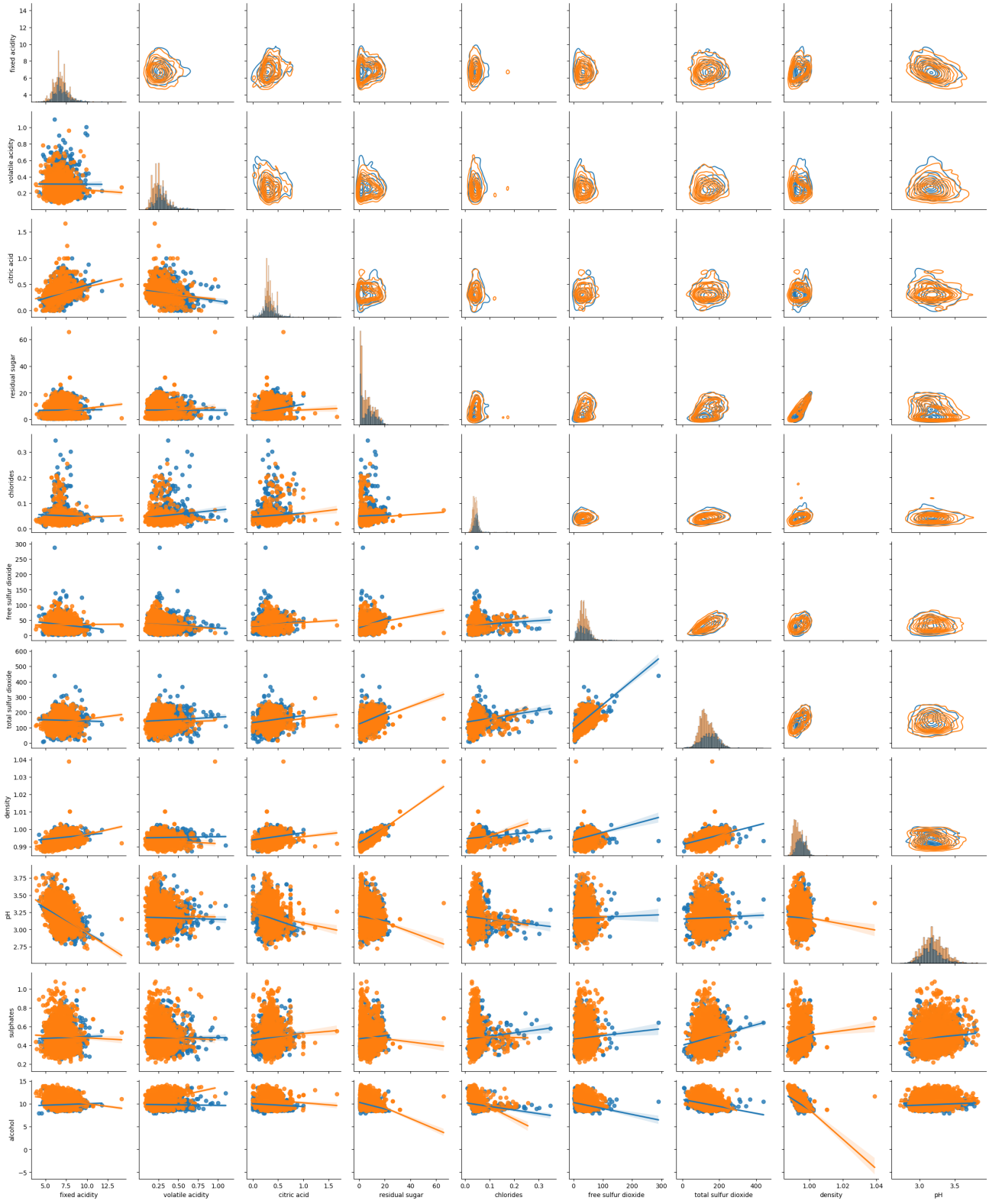
```
# Organizando dados para ficar apenas as features (variáveis contínuas)

white_wines = white_wines.drop(columns=['type', 'quality'])
features = white_wines.columns.drop('opinion')
features
```

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol'],
      dtype='object')
```

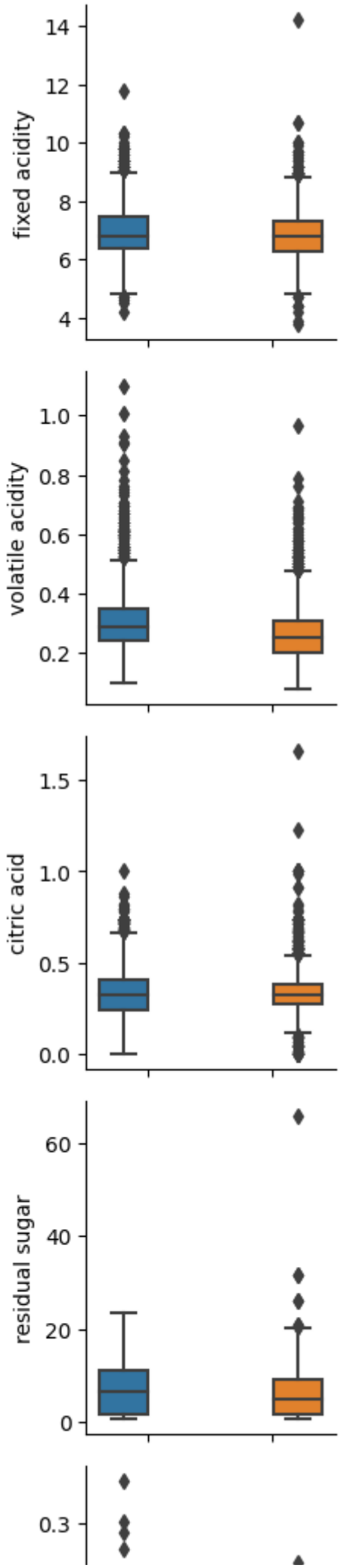
```
# Analisando as correlações entre as variáveis contínuas

fig, ax = plt.subplots(1, 1, figsize=(16, 8))
sns.heatmap(white_wines.corr(), vmin=-1, vmax=1, annot=True, ax=ax, cmap='vlag');
```

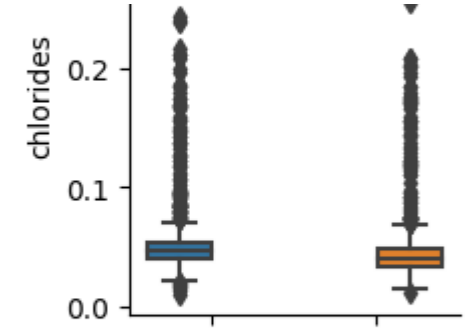


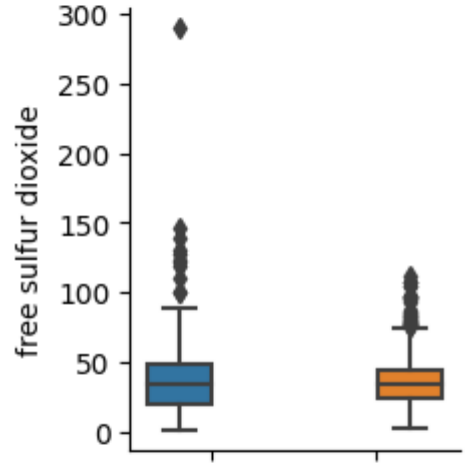

```
g = sns.PairGrid(white_wines, hue='opinion')
g.map_lower(sns.regplot)
g.map_diag(sns.histplot)
g.map_upper(sns.kdeplot);
```

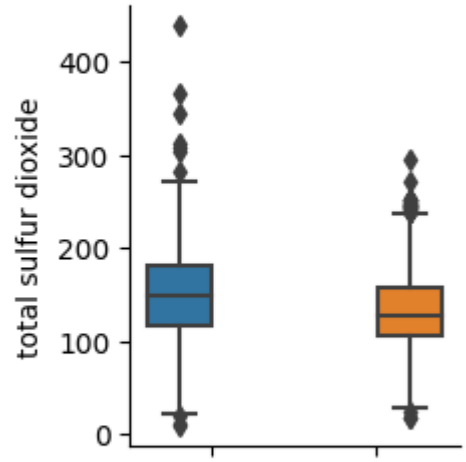


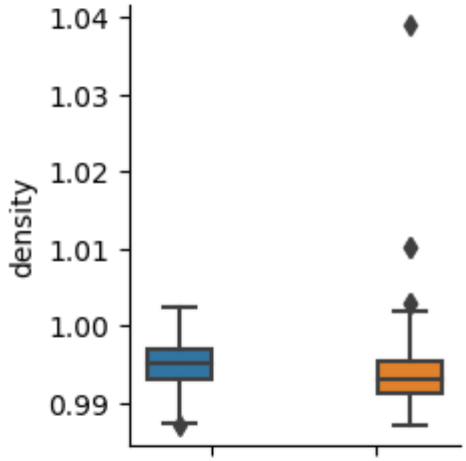

```
# Checando Outliers com Boxplot - não há nenhum muito gritante

g = sns.PairGrid(white_wines, hue='opinion', x_vars='opinion')
g.map(sns.boxplot);
```

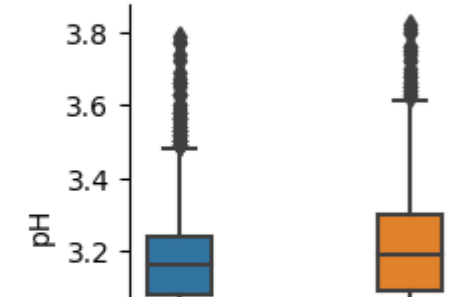

fixed acidity
14
12
10
8
6
4
volatile acidity
1.0
0.8
0.6
0.4
0.2
citric acid
1.5
1.0
0.5
0.0
residual sugar
60
40
20
0
0.3

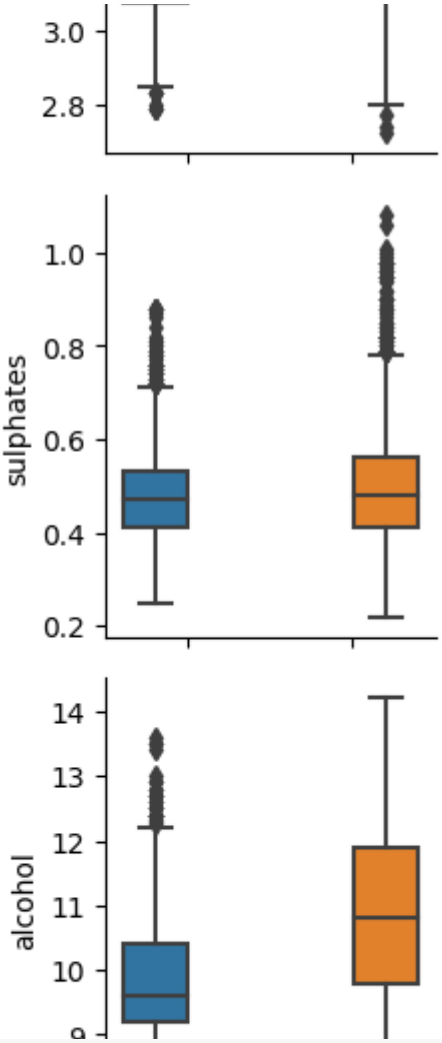

chlorides
0.2
0.1
0.0


free sulfur dioxide
300
250
200
150
100
50
0

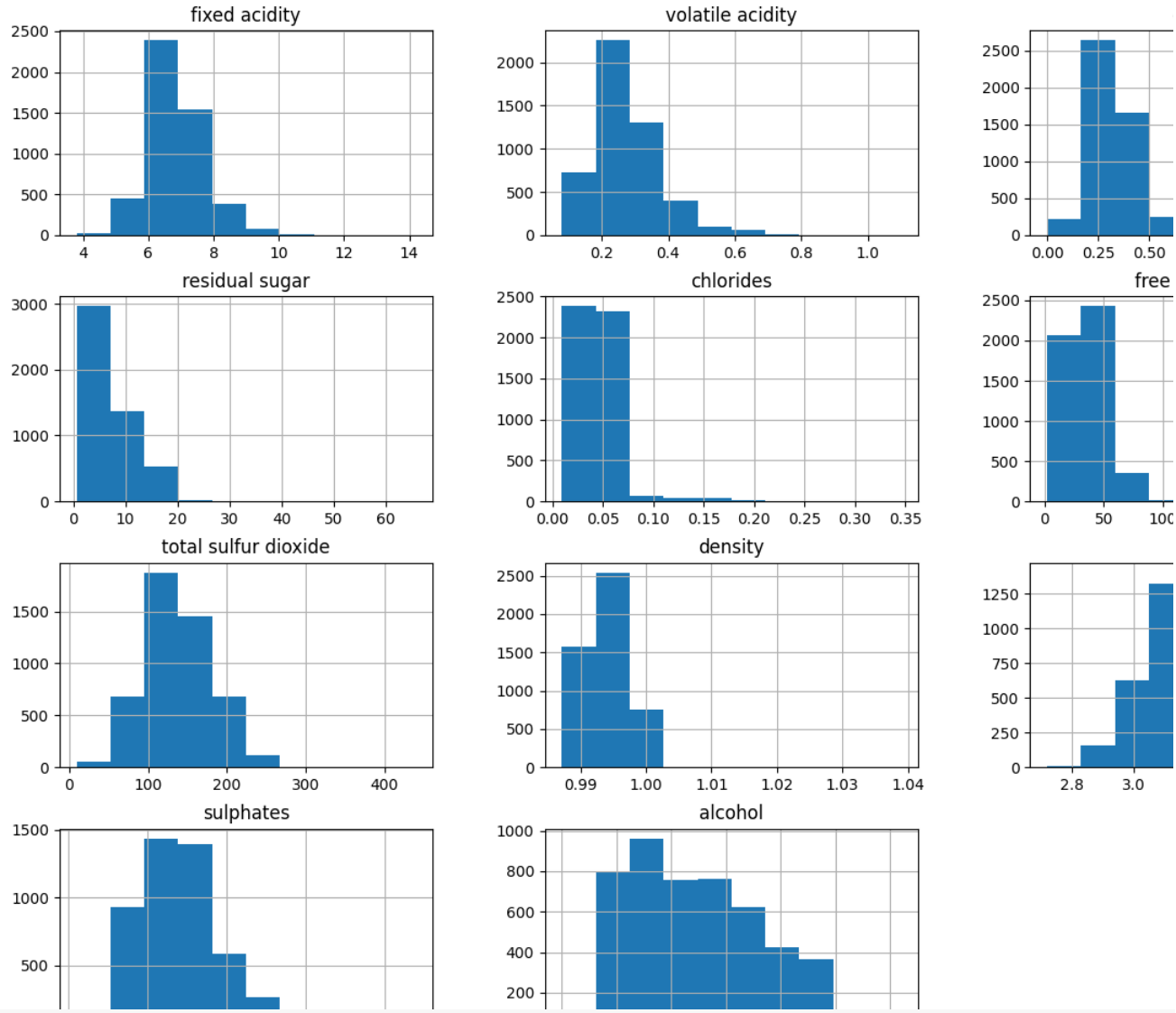

total sulfur dioxide
400
300
200
100
0


density
1.04
1.03
1.02
1.01
1.00
0.99

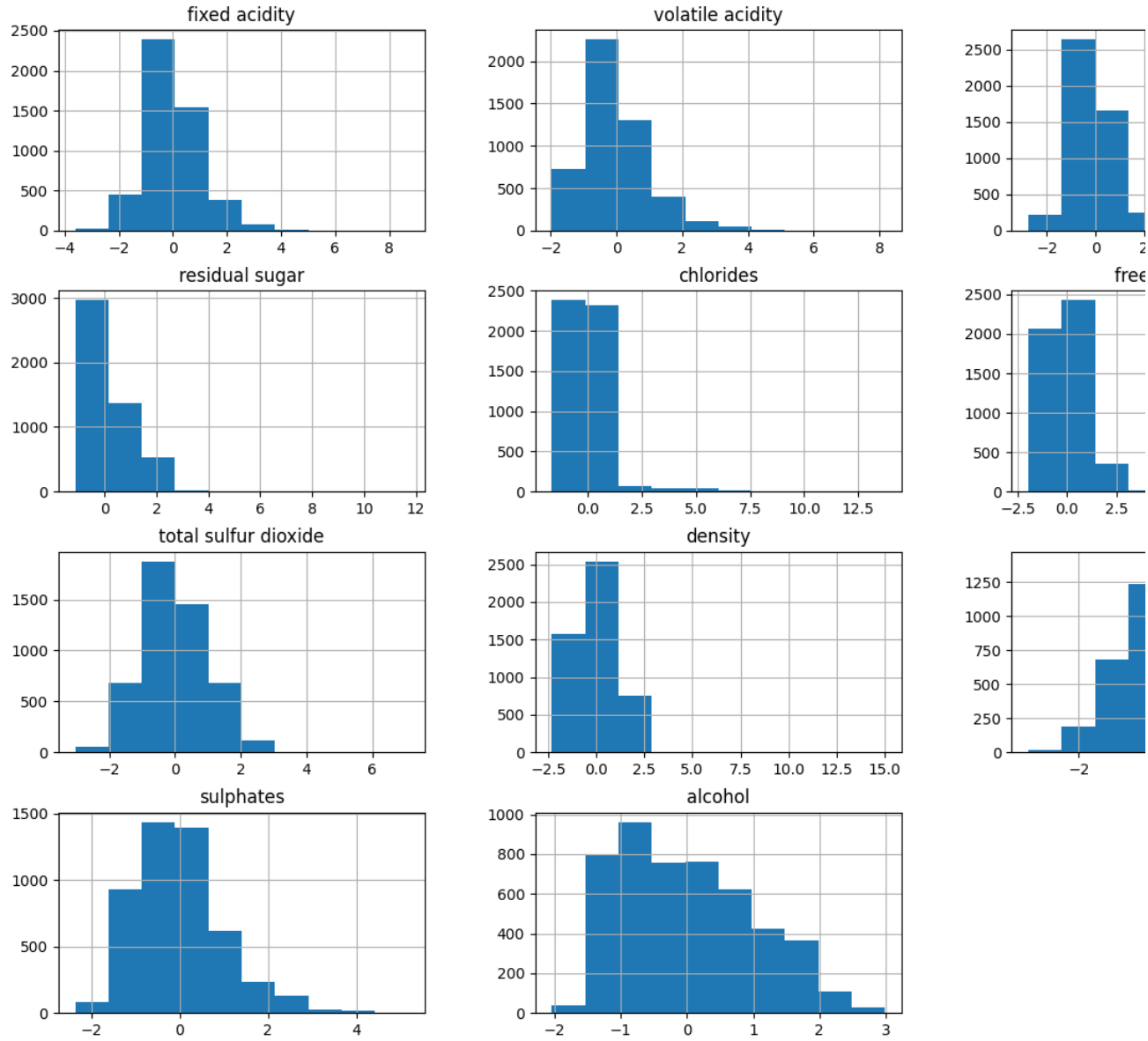

pH
3.8
3.6
3.4
3.2




```
# Analisando distribuição das variáveis com histogramas

white_wines[features].hist(figsize=(16, 12));
```



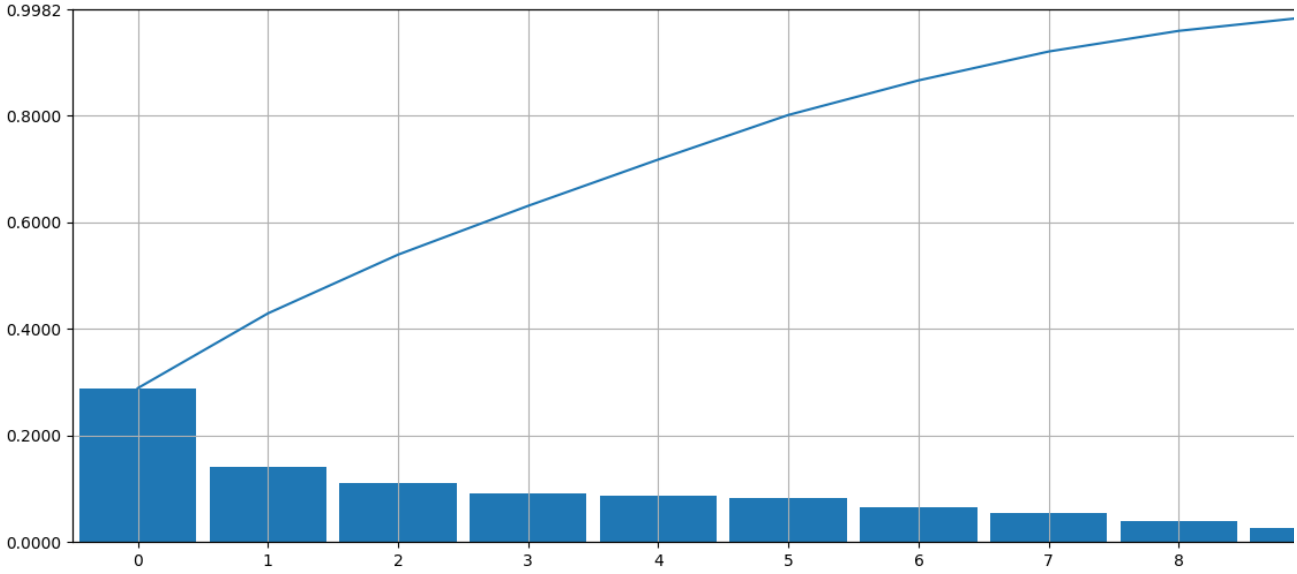

```
# Escalonando com StandardScaler (ss)

df_ss = white_wines.copy()
ss = StandardScaler()
df_ss[features] = ss.fit_transform(white_wines[features])
df_ss[features].hist(figsize=(16,12));
```




```
# Usando PCA / Pareto (Com 7 variáveis já ultrapassa 95%, poderia ser válido reduzir a dim

# !pip install rogeriopradoj-paretochart

from sklearn.decomposition import PCA
from paretochart.paretochart import pareto

pca= PCA(n_components=11)
pca.fit(df_ss)

fig, ax = plt.subplots(1, 1, figsize=(16, 6))
pareto(pca.explained_variance_ratio_)
ax.grid()

pd.DataFrame(pca.components_.T, columns=[f'PC{d}' for d in range(11)], index=df_ss.columns
```

```
C:\Users\Fabio\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra
  ax2.set_yticklabels(yt)
```

	PC0	PC1	PC2	PC3	PC4	PC5	PC6	
fixed acidity	0.156807	-0.586715	0.128982	0.032206	0.245284	0.105878	-0.200063	0.5
volatile acidity	0.009469	0.041054	-0.600492	0.262179	0.638486	-0.095564	0.272563	0.0
citric acid	0.142362	-0.338542	0.501410	0.177869	0.048582	-0.131000	0.704807	-0.1
residual sugar	0.424859	0.008972	-0.183148	-0.297854	0.013069	0.293808	0.212858	-0.4
chlorides	0.213394	-0.010190	-0.131626	0.682912	-0.338649	-0.401122	-0.079322	-0.1
free sulfur dioxide	0.297051	0.296314	0.288142	-0.282421	0.201516	-0.491878	-0.166277	-0.0
total sulfur dioxide	0.404966	0.245760	0.123673	-0.042399	0.302837	-0.270052	-0.064986	0.2
density	0.510587	0.004957	-0.117580	-0.038118	-0.092970	0.326720	0.108363	0.0
pH	-0.127702	0.582522	0.110817	0.109007	-0.129341	0.190603	0.423009	0.5
sulphates	0.042239	0.224481	0.411649	0.473217	0.371391	0.490529	-0.312507	-0.2
alcohol	-0.438655	-0.032491	0.114509	-0.126452	0.345399	-0.129015	0.130838	-0.2
opinion	-0.074605	0.016758	0.116184	-0.093178	0.005826	0.007963	0.000835	-0.1



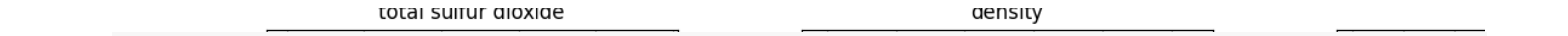

```
# Escalonando com PowerTransformer (pt) - Apply a power transform featurewise to make data

df_pt = white_wines.copy()
pt = PowerTransformer()
```

```
df_pt[features] = pt.fit_transform(white_wines[features])
df_pt[features].hist(figsize=(16,12));
```

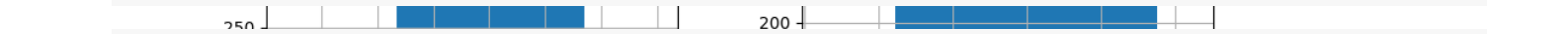
Agora defina o modelo de redes neural do tipo MLP que você irá utilizar. Essa é a fase de MODELAGEM: Quantas camadas? Quantos neurônios na camada de entrada? Quantos neurônios na camada de saída? Justifique. Quantos números de neurônios na camada intermediária? Como esse número foi escolhido?

RESPONDENDO:

Como são 11 features, serão 11 neurônios na primeira camada (camada de entrada). Na última camada, sempre é apenas um neurônio, pois é assim que funciona toda rede neural. O número de neurônios na camada intermediária escolhemos usando o gráfico abaixo, de forma a maximizar o resultado de F1 Score mas sem overtraining: 26 neurônios, conforme veremos abaixo.




```
# Com todos os dados preparados, iniciamos redes neurais

from sklearn.model_selection import (
    train_test_split,
    StratifiedKFold)
from sklearn.metrics import (f1_score,
                             ConfusionMatrixDisplay,
                             classification_report)

# Neural network
from sklearn.neural_network import MLPClassifier
```




```
wines['quality'].unique()
```

```
array([6, 5, 7, 8, 4, 3, 9], dtype=int64)
```

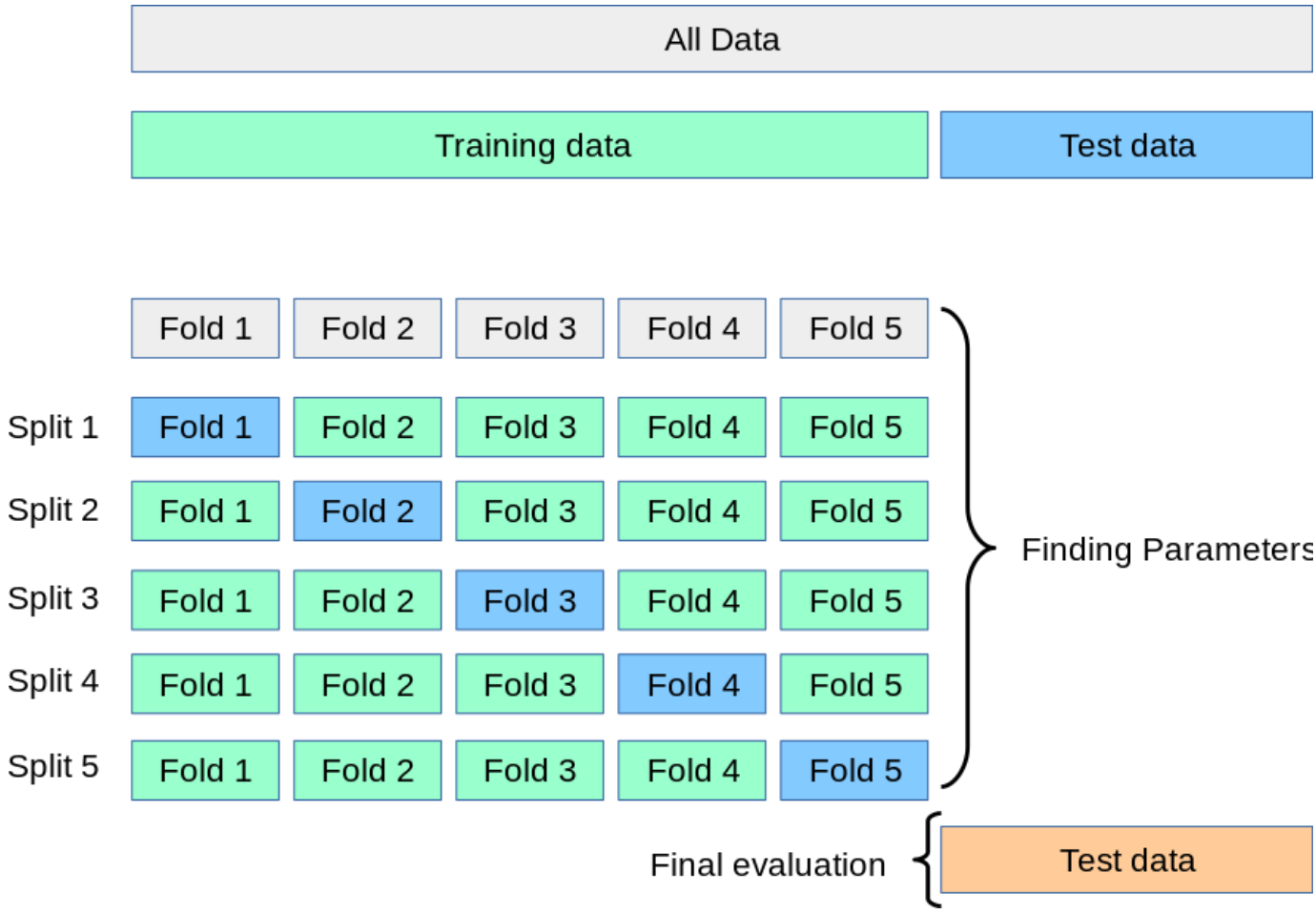
```
wines[features]
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH

```
wines['opinion']
```

```
0       1
1       1
2       1
3       1
4       1
       ..
6491    1
6492    0
6494    1
6495    0
6496    1
Name: opinion, Length: 6463, dtype: int64
```

Treinamento com validação cruzada




```
X = wines[features].values
y = wines['opinion'].values

X_train_cv, X_test, y_train_cv, y_test = train_test_split(X,
                                                          y,
```

```
                                                    test_size=0.2, # 20 % da base
                                                    random_state=42,
                                                    stratify=y)
```

```
X_train_cv
```

```
     array([[ 7.3  ,  0.17 ,  0.23 , ...,  3.36 ,  0.54 , 10.   ],
            [10.9  ,  0.32 ,  0.52 , ...,  3.28 ,  0.77 , 11.5  ],
            [ 6.2  ,  0.26 ,  0.32 , ...,  3.31 ,  0.61 ,  9.4  ],
            ...,
            [ 6.1  ,  0.22 ,  0.49 , ...,  3.3  ,  0.46 ,  9.6  ],
            [ 8.6  ,  0.265,  0.36 , ...,  2.95 ,  0.36 , 11.4  ],
            [ 6.3  ,  0.25 ,  0.23 , ...,  3.14 ,  0.35 ,  9.7  ]])
```

```
mlp = MLPClassifier(random_state=42)
```

```
def train(X, y, model_klass, n_splits=5, n_init=1, **kwargs):
    cv = StratifiedKFold(n_splits=n_splits)
    f1_score_val_list = []
    f1_score_train_list = []
    model_list =[]
    scaler_list = []
    # Validação cruzada só em Training Data
    for fold, (train_idx, val_idx) in enumerate(cv.split(X, y)):
        X_train = X[train_idx, :]
        y_train = y[train_idx]
        X_val = X[val_idx, :]
        y_val = y[val_idx]

        # Escala
        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_val_scaled = scaler.transform(X_val)

        scaler_list.append(scaler)
        # TO DO - Fazer a PCA como opcional
        # Treino
        model = None
        f1_score_val = 0.
        for idx in range(n_init):
            _model = model_klass(**kwargs)
            _model.fit(X_train_scaled, y_train)
            _y_pred = _model.predict(X_train_scaled)
            _y_pred_val = _model.predict(X_val_scaled)
            _f1_score_val = f1_score(y_val, _y_pred_val)
            if _f1_score_val > f1_score_val:
                y_pred_val = _y_pred_val
                y_pred = _y_pred
                model = _model
        print(f"======================== FOLD {fold} ========================")
        print(f"Meu resultado para treino de F1-Score é {f1_score(y_train, y_pred):.2}")
        print(f"Meu resultado para validação de F1-Score é {f1_score(y_val, y_pred_val):.2
```

```
        f1_score_val_list.append(f1_score(y_val, y_pred_val))
        f1_score_train_list.append(f1_score(y_train, y_pred))
        model_list.append(model)
    print()
    print()
    mean_val = np.mean(f1_score_val_list)
    std_val = np.std(f1_score_val_list)
    print(f"Meu resultado de F1-Score Médio de treino é \
          {np.mean(f1_score_train_list): .2} +- {np.std(f1_score_train_list): .2} ")
    print(f"Meu resultado de F1-Score Médio de validação é {mean_val: .2} +- {std_val: .2}
    print()

    best_model_idx = np.argmax(f1_score_val_list)
    print(f"Meu melhor fold é: {best_model_idx} ")
    best_model = model_list[best_model_idx]
    best_scaler = scaler_list[best_model_idx]
    return best_model, mean_val, std_val, best_scaler
```

Qual será o algoritmo utilizado no treinamento? Qual a função de ativação será utilizada em cada camada? Justifique. Qual a função de otimização será utilizada neste treinamento? Quais serão as figuras de métrica que serão usadas na avaliação do modelo. Justifique. Apresente os seguintes resultados: A matriz de confusão do problema O(s) histograma(s) da(s) saída(s) da rede neural Os valores das figuras de métrica utilizadas. Caso tenha utilizado validação cruzada no treino, apresente as incertezas para cada um dos resultados anteriores. Avalie os resultados, dando sua interpretação de acordo com a compreensão da natureza do problema proposto. Essa é a fase de AVALIAÇÃO DO MODELO.

RESPONDENDO:

A função logística, ou sigmoide, pode ser uma escolha adequada para um problema de classificação binária com 26 neurônios. A função logística mapeia os valores de entrada para um intervalo entre 0 e 1, o que permite interpretar a saída como a probabilidade de pertencer à classe positiva. Porém, atualmente a tangente hiperbólica tende a gerar melhores resultados (também classificação binária), então alteramos para usá-la.

Em resumo, a função logística (sigmóide) ou a tangente hiperbólica pode ser usada como função de ativação para os neurônios de saída em um problema de classificação binária com 26 neurônios, mas é importante considerar a utilização de outras funções de ativação, como a função ReLU, para as camadas ocultas, dependendo da complexidade da sua rede neural.

O algoritmo de treinamento é sempre o backpropagation (retroalimentação).

As figuras de mérito são acurácia, precisão, f1-score, sensibilidade, área sobre a curva ROC (AUC) - figuras usadas para avaliar o modelo. Neste caso, utilizamos as abaixo:

Model

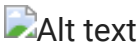
Neural Network

```
nn_results = []
for neurons in range(1, 30):
    nn_model, nn_mean_val, nn_std_val, nn_scaler = train(X_train_cv, y_train_cv,
                                                         MLPClassifier,
                                                         hidden_layer_sizes=(neurons,),
                                                         n_init=10,
                                                         max_iter=10000,
                                                         activation="tanh", # Mudando func
                                                                            # para tangent
                                                         alpha=1e-5,
                                                         tol=1e-3,
                                                         learning_rate_init=.3,
                                                         solver='sgd')

    nn_results.append((nn_model, nn_mean_val, nn_std_val, nn_scaler))
```

```
======================== FOLD 0 ========================
Meu resultado para treino de F1-Score é 0.79
Meu resultado para validação de F1-Score é 0.77
======================== FOLD 1 ========================
Meu resultado para treino de F1-Score é 0.81
Meu resultado para validação de F1-Score é 0.8
======================== FOLD 2 ========================
Meu resultado para treino de F1-Score é 0.8
Meu resultado para validação de F1-Score é 0.8
======================== FOLD 3 ========================
Meu resultado para treino de F1-Score é 0.78
Meu resultado para validação de F1-Score é 0.79
======================== FOLD 4 ========================
Meu resultado para treino de F1-Score é 0.79
Meu resultado para validação de F1-Score é 0.8


Meu resultado de F1-Score Médio de treino é            0.79 +-  0.0086
Meu resultado de F1-Score Médio de validação é  0.79 +-  0.011

Meu melhor fold é: 4
======================== FOLD 0 ========================
Meu resultado para treino de F1-Score é 0.82
Meu resultado para validação de F1-Score é 0.81
======================== FOLD 1 ========================
Meu resultado para treino de F1-Score é 0.81
Meu resultado para validação de F1-Score é 0.8
======================== FOLD 2 ========================
Meu resultado para treino de F1-Score é 0.81
Meu resultado para validação de F1-Score é 0.81
======================== FOLD 3 ========================
Meu resultado para treino de F1-Score é 0.81
Meu resultado para validação de F1-Score é 0.81
======================== FOLD 4 ========================
Meu resultado para treino de F1-Score é 0.81
Meu resultado para validação de F1-Score é 0.81
```

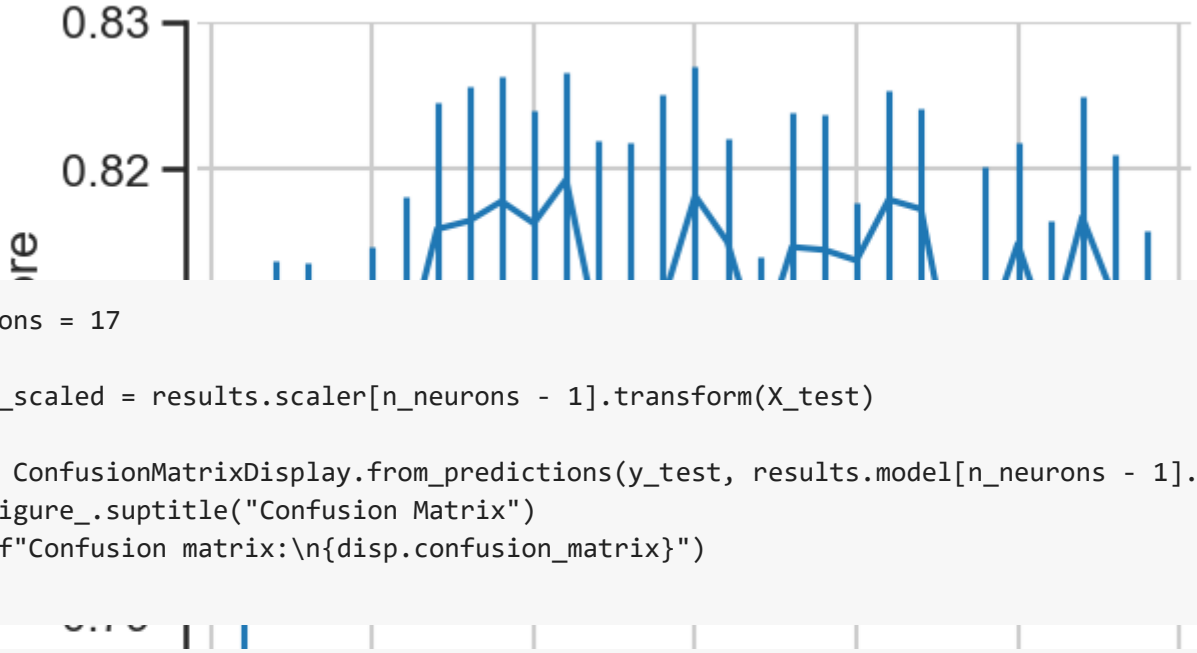
```
Meu resultado de F1-Score Médio de treino é            0.81 +-  0.0038
Meu resultado de F1-Score Médio de validação é  0.81 +-  0.0059

Meu melhor fold é: 4
======================== FOLD 0 ========================
Meu resultado para treino de F1-Score é 0.81
Meu resultado para validação de F1-Score é 0.8
======================== FOLD 1 ========================
Meu resultado para treino de F1-Score é 0.81
Meu resultado para validação de F1-Score é 0.8
======================== FOLD 2 ========================
Meu resultado para treino de F1-Score é 0.81
Meu resultado para validação de F1-Score é 0.81
======================== FOLD 3 ========================
Meu resultado para treino de F1-Score é 0.82
Meu resultado para validação de F1-Score é 0.82
======================== FOLD 4 ========================
Meu resultado para treino de F1-Score é 0.82
Meu resultado para validação de F1-Score é 0.8
```

```
# Apenas para voltar ao nn_results não como lista:

nn_results = pd.DataFrame(nn_results, columns = ['model', 'mean f1', 'std f1', 'scaler'])
```




```
sns.set_style("ticks")
sns.set_context("talk")
plt.errorbar(range(1, 30), nn_results["mean f1"], nn_results["std f1"])
plt.grid(True)
sns.despine(offset=5)
plt.ylim([ 0.78 , 0.83])
plt.ylabel("Mean F1 Score ")
plt.xlabel("N Neurons")
```

```
Text(0.5, 0, 'N Neurons')
```

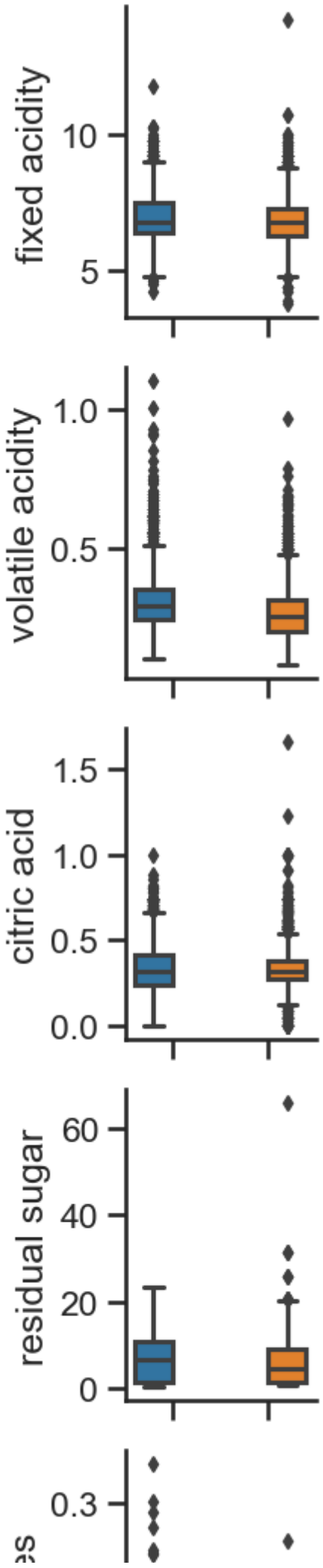


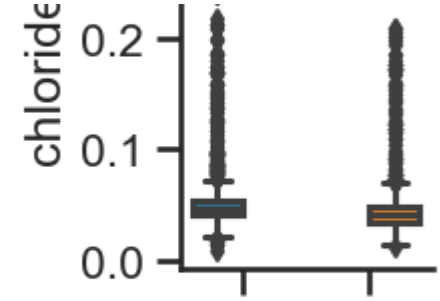

```
n_neurons = 17

X_test_scaled = results.scaler[n_neurons - 1].transform(X_test)

disp = ConfusionMatrixDisplay.from_predictions(y_test, results.model[n_neurons - 1].predic
disp.figure_.suptitle("Confusion Matrix")
print(f"Confusion matrix:\n{disp.confusion_matrix}")
```

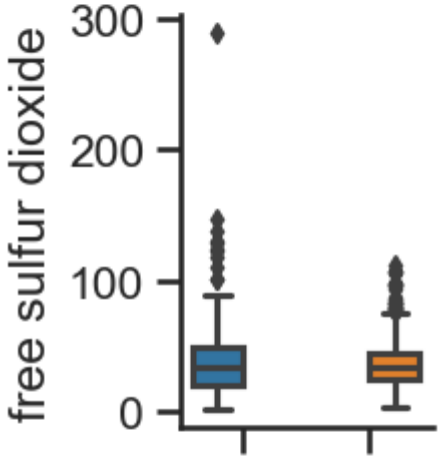
```
n_neurons = 20

X_test_scaled = nn_results['scaler'][n_neurons - 1].transform(X_test)

disp = ConfusionMatrixDisplay.from_predictions(y_test, nn_results.model[n_neurons - 1].pre
disp.figure_.suptitle("Confusion Matrix")
print(f"Confusion matrix:\n{disp.confusion_matrix}")
```

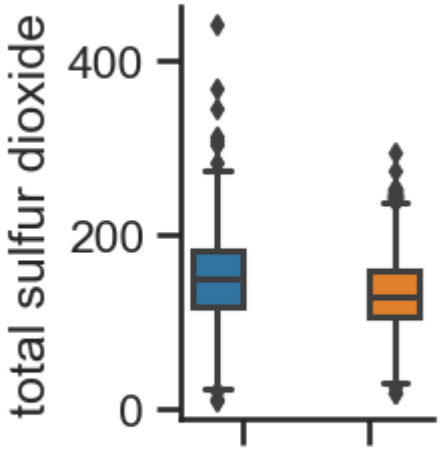
```
Confusion matrix:
[[ 511  211]
```

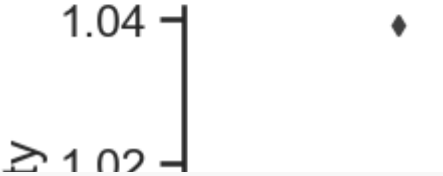
```
g = sns.PairGrid(white_wines, hue='opinion', x_vars='opinion')
g.map(sns.boxplot);
```


fixed acidity
10
5
volatile acidity
1.0
0.5
citric acid
1.5
1.0
0.5
0.0
residual sugar
60
40
20
0
0.3










```
print(classification_report(y_test, results.model[n_neurons - 1].predict(X_test_scaled)))
```

```
              precision    recall  f1-score   support

           0       0.71      0.64      0.67       475
           1       0.80      0.84      0.82       818

    accuracy                           0.77      1293
   macro avg       0.75      0.74      0.75      1293
weighted avg       0.77      0.77      0.77      1293
```

Regressão Logística

```
from sklearn.linear_model import LogisticRegression

logit, logit_mean_val, logit_std_val, logit_scaler = train(X_train_cv, y_train_cv, Logisti
```

```
======================== FOLD 0 ========================
Meu resultado para treino de F1-Score é 0.72
Meu resultado para validação de F1-Score é 0.7
======================== FOLD 1 ========================
Meu resultado para treino de F1-Score é 0.72
Meu resultado para validação de F1-Score é 0.71
======================== FOLD 2 ========================
Meu resultado para treino de F1-Score é 0.71
Meu resultado para validação de F1-Score é 0.71
======================== FOLD 3 ========================
Meu resultado para treino de F1-Score é 0.71
Meu resultado para validação de F1-Score é 0.72
======================== FOLD 4 ========================
Meu resultado para treino de F1-Score é 0.71
Meu resultado para validação de F1-Score é 0.71


Meu resultado de F1-Score Médio de treino é  0.71 +-  0.0039
Meu resultado de F1-Score Médio de validação é  0.71 +-  0.0053

Meu melhor fold é: 3
```

```
pd.DataFrame(logit.predict_proba(X_test_scaled))
```

	0	1
0	0.313238	0.686762
1	0.052954	0.947046
2	0.073718	0.926282
3	0.528420	0.471580
4	0.211477	0.788523
...	...	...
1288	0.109647	0.890353
1289	0.263083	0.736917
1290	0.267792	0.732208
1291	0.650066	0.349934
1292	0.155907	0.844093

1293 rows × 2 columns

SVM

```
svm_rbf_model, svm_rbf_mean_val, svm_rbf_std_val, svm_rbf_scaler = train(X_train_cv,
                                                                         y_train_cv,
                                                                         SVC,
                                                                         gamma = 'auto',
                                                                         C= 1,
                                                                         kernel= 'rbf')
```

```
======================== FOLD 0 ========================
Meu resultado para treino de F1-Score é 0.78
Meu resultado para validação de F1-Score é 0.74
======================== FOLD 1 ========================
Meu resultado para treino de F1-Score é 0.78
Meu resultado para validação de F1-Score é 0.76
======================== FOLD 2 ========================
Meu resultado para treino de F1-Score é 0.78
Meu resultado para validação de F1-Score é 0.75
======================== FOLD 3 ========================
Meu resultado para treino de F1-Score é 0.79
Meu resultado para validação de F1-Score é 0.75
======================== FOLD 4 ========================
Meu resultado para treino de F1-Score é 0.78
Meu resultado para validação de F1-Score é 0.73


Meu resultado de F1-Score Médio de treino é  0.78 +-  0.003
Meu resultado de F1-Score Médio de validação é  0.75 +-  0.01

Meu melhor fold é: 1
```

```
svm_poly_3_model, svm_poly_3_mean_val, svm_poly_3_std_val, svm_poly_3_scaler = train(X_tra
                                                                                     y_tra
                                                                                     SVC,
                                                                                     gamma
                                                                                     C=1,
                                                                                     degre
                                                                                     kerne

svm_poly_5_model, svm_poly_5_mean_val, svm_poly_5_std_val, svm_poly_5_scaler = train(X_tra
                                                                                     y_tra
                                                                                     SVC,
                                                                                     gamma
                                                                                     C=1,
                                                                                     degre
                                                                                     kerne

svm_poly_10_model, svm_poly_10_mean_val, svm_poly_10_std_val, svm_poly_10_scaler = train(X
                                                                                        y
                                                                                        S
                                                                                        g
                                                                                        C
                                                                                        d
                                                                                        k
```

```
======================== FOLD 0 ========================
Meu resultado para treino de F1-Score é 0.83
Meu resultado para validação de F1-Score é 0.81
```

```
======================== FOLD 1 ========================
Meu resultado para treino de F1-Score é 0.83
Meu resultado para validação de F1-Score é 0.82
======================== FOLD 2 ========================
Meu resultado para treino de F1-Score é 0.83
Meu resultado para validação de F1-Score é 0.81
======================== FOLD 3 ========================
Meu resultado para treino de F1-Score é 0.83
Meu resultado para validação de F1-Score é 0.82
======================== FOLD 4 ========================
Meu resultado para treino de F1-Score é 0.83
Meu resultado para validação de F1-Score é 0.81


Meu resultado de F1-Score Médio de treino é  0.83 +-  0.002
Meu resultado de F1-Score Médio de validação é  0.81 +-  0.0042

Meu melhor fold é: 3
======================== FOLD 0 ========================
Meu resultado para treino de F1-Score é 0.84
Meu resultado para validação de F1-Score é 0.81
======================== FOLD 1 ========================
Meu resultado para treino de F1-Score é 0.84
Meu resultado para validação de F1-Score é 0.8
======================== FOLD 2 ========================
Meu resultado para treino de F1-Score é 0.83
Meu resultado para validação de F1-Score é 0.8
======================== FOLD 3 ========================
Meu resultado para treino de F1-Score é 0.84
Meu resultado para validação de F1-Score é 0.81
======================== FOLD 4 ========================
Meu resultado para treino de F1-Score é 0.83
Meu resultado para validação de F1-Score é 0.8


Meu resultado de F1-Score Médio de treino é  0.84 +-  0.00093
Meu resultado de F1-Score Médio de validação é  0.8 +-  0.0041

Meu melhor fold é: 0
======================== FOLD 0 ========================
Meu resultado para treino de F1-Score é 0.85
Meu resultado para validação de F1-Score é 0.8
======================== FOLD 1 ========================
Meu resultado para treino de F1-Score é 0.84
Meu resultado para validação de F1-Score é 0.79
======================== FOLD 2 ========================
Meu resultado para treino de F1-Score é 0.85
Meu resultado para validação de F1-Score é 0.79
======================== FOLD 3 ========================
Meu resultado para treino de F1-Score é 0.84
Meu resultado para validação de F1-Score é 0.8
======================== FOLD 4 ========================
Meu resultado para treino de F1-Score é 0.85
Meu resultado para validação de F1-Score é 0.79
```

Decision Tree

```
tree_model, tree_mean_val, tree_std_val, tree_scaler  = train(X_train_cv, y_train_cv, Deci
```

```
======================== FOLD 0 ========================
Meu resultado para treino de F1-Score é 1.0
Meu resultado para validação de F1-Score é 0.73
======================== FOLD 1 ========================
Meu resultado para treino de F1-Score é 1.0
Meu resultado para validação de F1-Score é 0.73
======================== FOLD 2 ========================
Meu resultado para treino de F1-Score é 1.0
Meu resultado para validação de F1-Score é 0.75
======================== FOLD 3 ========================
Meu resultado para treino de F1-Score é 1.0
Meu resultado para validação de F1-Score é 0.76
======================== FOLD 4 ========================
Meu resultado para treino de F1-Score é 1.0
Meu resultado para validação de F1-Score é 0.75


Meu resultado de F1-Score Médio de treino é  1.0 +-  0.0
Meu resultado de F1-Score Médio de validação é  0.74 +-  0.01

Meu melhor fold é: 3
```

```
tree_50_leaf_model, tree_50_leaf_mean_val, tree_50_leaf_std_val, tree_50_leaf_scaler  = tr
```

```
======================== FOLD 0 ========================
Meu resultado para treino de F1-Score é 0.83
Meu resultado para validação de F1-Score é 0.79
======================== FOLD 1 ========================
Meu resultado para treino de F1-Score é 0.82
Meu resultado para validação de F1-Score é 0.79
======================== FOLD 2 ========================
Meu resultado para treino de F1-Score é 0.82
Meu resultado para validação de F1-Score é 0.81
======================== FOLD 3 ========================
Meu resultado para treino de F1-Score é 0.82
Meu resultado para validação de F1-Score é 0.8
======================== FOLD 4 ========================
Meu resultado para treino de F1-Score é 0.83
Meu resultado para validação de F1-Score é 0.8


Meu resultado de F1-Score Médio de treino é  0.83 +-  0.0038
Meu resultado de F1-Score Médio de validação é  0.8 +-  0.009

Meu melhor fold é: 2
```

Overall Results

```
results = [
    ('Logistic Regression', logit_mean_val, logit_std_val),
```

```
    ('Decision Tree', tree_mean_val, tree_std_val),
    ('Decision Tree - min. 50 leafs', tree_50_leaf_mean_val, tree_50_leaf_std_val),
    ('SVM - RBF', svm_rbf_mean_val, svm_rbf_std_val),
    ('SVM Poly 3', svm_poly_3_mean_val, svm_poly_3_std_val),
    ('SVM Poly 5', svm_poly_5_mean_val, svm_poly_5_std_val),
    ('SVM Poly 10', svm_poly_10_mean_val, svm_poly_10_std_val),
] + nn_results

results = pd.DataFrame(results, columns=['Model', 'Validation F1', 'Validation F1 deviatio
results.set_index('Model', inplace=True)
```

```
results
```

Model	Validation F1	Validation F1 deviation
Logistic Regression	0.709616	0.005322
Decision Tree	0.743511	0.010492
Decision Tree - min. 50 leafs	0.800908	0.009018
SVM - RBF	0.746212	0.010471
SVM Poly 3	0.812797	0.004168
SVM Poly 5	0.803840	0.004085
SVM Poly 10	0.794429	0.002795
Neural Network (1 neurons)	0.799620	0.013773
Neural Network (2 neurons)	0.805203	0.005840
Neural Network (3 neurons)	0.813723	0.013743
Neural Network (4 neurons)	0.811075	0.009668

- Melhor resultado com 26 neurônios (maior valor e menor erro)

Neural Network (7 neurons)	0.813372	0.010045
Neural Network (8 neurons)	0.812607	0.013835
Neural Network (9 neurons)	0.814640	0.008231
Neural Network (10 neurons)	0.817285	0.005418
Neural Network (11 neurons)	0.813368	0.009612
Neural Network (12 neurons)	0.817202	0.009531
Neural Network (13 neurons)	0.814748	0.011426
Neural Network (14 neurons)	0.817412	0.009867
Neural Network (15 neurons)	0.813715	0.011084