Relatório do Projeto - Classificação de Qualidade de Vinhos.

Equipe 7 - Ednelson Luan Lourenço Cavalcanti, José Danilo Silva do Carmo e Rafael Rodrigues da Silva.

1. Introdução

O projeto visou a implementação e avaliação de diferentes classificadores, incluindo Árvores de Decisão, Naive Bayes, Regressão Logística e K-Vizinhos, para a tarefa de classificação da qualidade do vinho com base em um conjunto de dados específico. O desafio consistiu em maximizar métricas de desempenho, como precision, recall e f1-score, para cada classificador.

2. Metodologia

2.1 Importação das bibliotecas e dataset:

Para iniciarmos o desenvolvimento do projeto utilizamos algumas bibliotecas importantes, o numpy para cálculos matemáticos, o pandas para a leitura do dataset, e ambos o seaborn e o matplotlib para a plotagem de gráficos.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
from google.colab import drive
drive.mount('/content/drive')
BASE_PATH = "/content/drive/MyDrive"
base_wine = pd.read_csv(f"{BASE_PATH}/winequality.csv")
base_wine.describe()
```

2.2 Tratamento inicial:

Inicialmente verificamos se é preciso algum tratamento dos dados do dataset. Primeiro vemos se existe alguma feature com valor nulo (o que não ocorre) e logo em seguida (através de testes posteriores) vemos que a feature “alcohol” possui valores inconsistentes (números com muitas casas que eram vistos como string), e como teve um baixo n° de amostras com essa anomalia, decidimos tirar elas do dataset.

```
base_wine.isnull().any().any()

def is_float(x):
    try:
        float(x)
        return True
    except:
        return False

base_wine = base_wine[base_wine["alcohol"].apply(is_float)]
```

2.3 Escolha das Features e Target:

A priori nosso objetivo era que, com o auxílio dos classificadores, pudéssemos prever a nota exata de cada vinho na etapa de testes (Figura 1), mas houve uma acurácia muito baixa deste método, então partimos para o princípio de dividir (Figura 2) o vinho como sendo “bom” (acima de qualidade 5) ou “ruim” (com qualidade 5 ou abaixo).

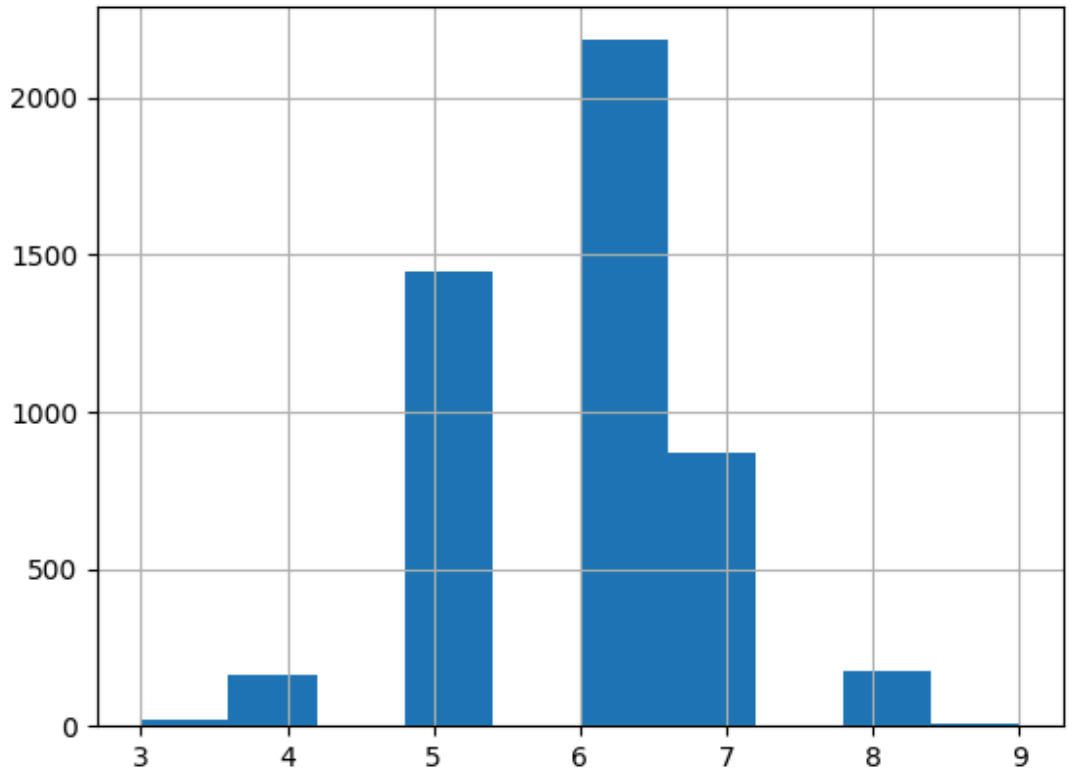

Figura 1

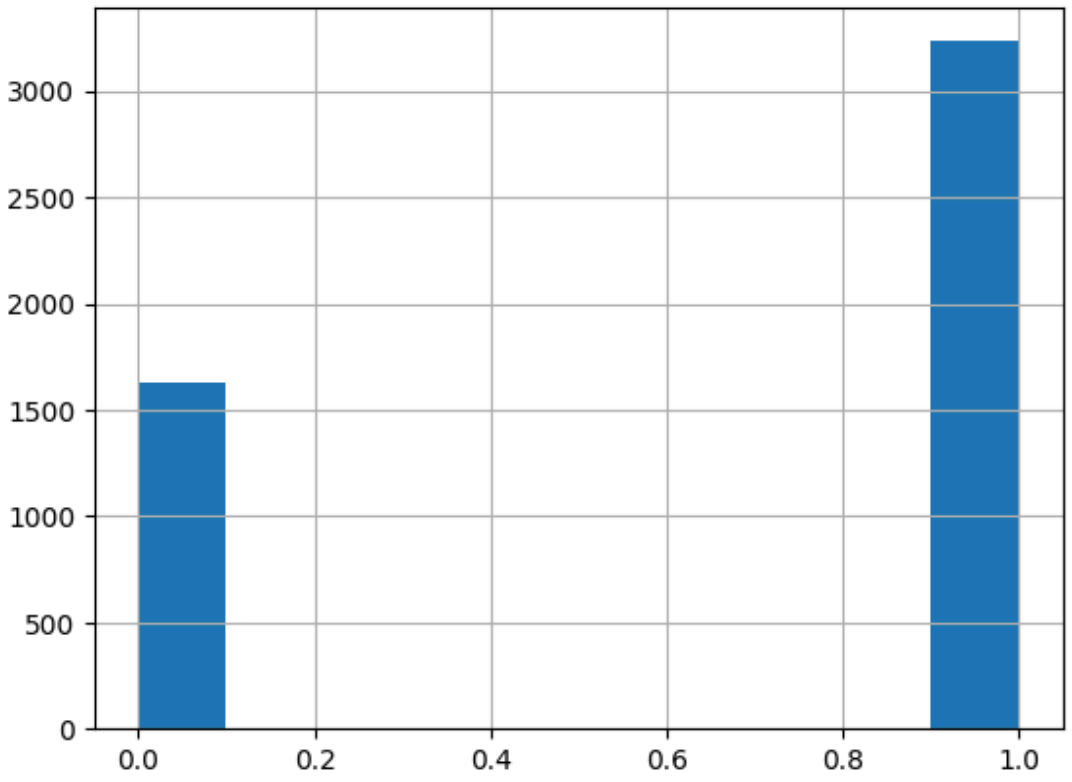

Figura 2

Após isso, a proporção entre vinhos bons e ruins ficaram aproximadamente 2:1, sendo necessário a utilização do SMOTE (Synthetic Minority Over-sampling Technique) para ajustar a relação entre as classes, sendo criados novos exemplos da classe minoritária, tornando possível que a regressão logística consiga convergir. Foi preciso também normalizar os dados, uma vez que os valores das features variam entre 1 e 999, para isso utilizamos o StandardScaler que remove a média e ajusta os dados para a variância unitária. Ou seja, para o um valor X é calculado:

$$z = \frac{(x - u)}{s}$$

Figura 3

Onde **u** é a média das amostras de treinamento (**x**) e **s** é o desvio padrão das amostras de treinamento. Utilizamos também a plotagem na representação de violino (Figura 4) para identificar features que poderiam ser removidas do dataset, caso fossem constantes. Que neste caso se observou que não há features com valores constantes.

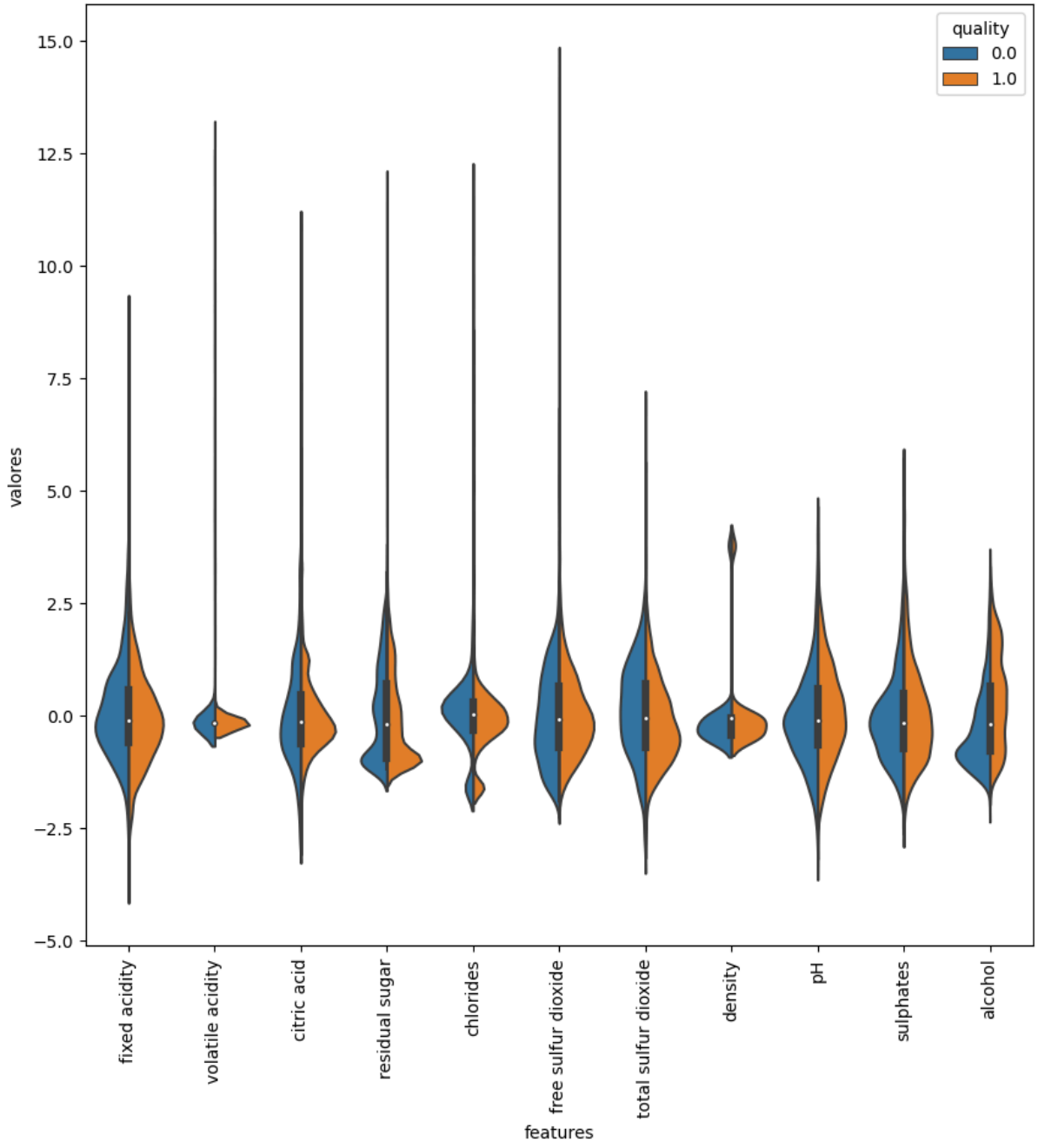

Figura 4

Assim como, utilizamos a matriz de correlação que checa se algum feature é bastante correlacionado com outra (Figura 5), onde podemos observar que a maior correlação entre as features foi de 60%, o que não achamos alta o suficiente para justificar a retirada de alguma delas, sendo assim, permanecendo todas as features iniciais.

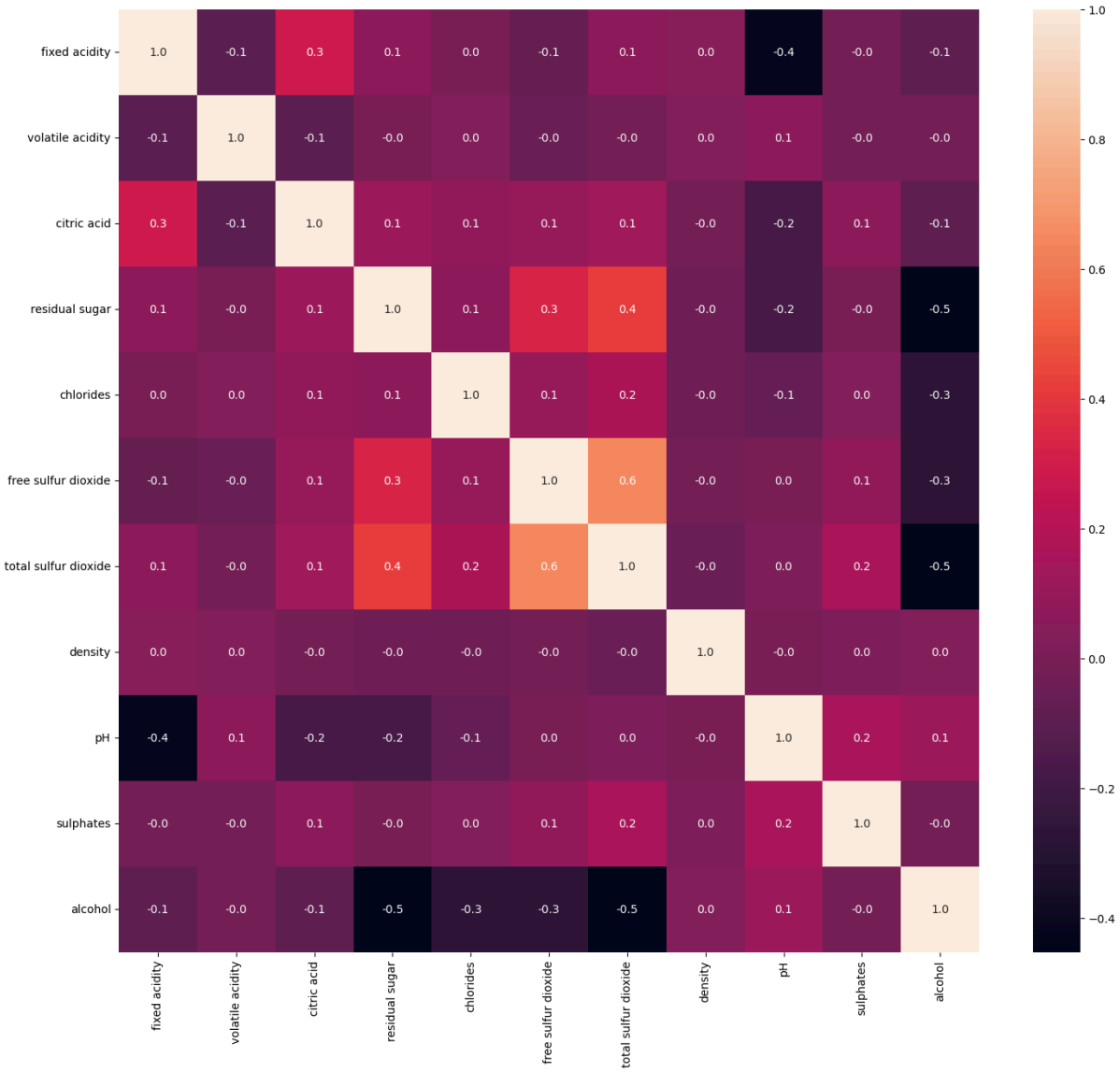

Figura 5

Posterior a isto, dividimos o nosso conjunto de dados em 25% para teste/validação e 75% para treinamento e utilizamos o classificador Dummy como modelo base para predição, utilizando como benchmark, considerando a estratégia "mais frequente". Nele, teremos um score base, que servirá para compararmos com os outros modelos, que foi de 49%.

3. Modelos

3.1 Árvore de Decisão:

Este algoritmo é chamado de árvore de decisão pois o modelo se divide em forma de árvore, iniciando com um nó inicial que se divide formando ramos (possíveis caminhos).

O GridSearchCV nos permite testar o modelo (neste caso o DecisionTreeClassifier) com os mais diversos parâmetros, devolvendo os

melhores parâmetros, com base na validação cruzada de tamanho 5, que será explicada no fim do projeto. Os parâmetros testados são:

- ***max_depth***: A profundidade máxima da árvore.
- ***min_samples_split***: Número mínimo de amostras para dividir um nó.
- ***min_samples_leaf***: Número mínimo de amostras para ser uma folha.
- ***criterion***: A função para medir a qualidade e uma divisão, podendo ser a medida de impureza "Gini", que mede a frequência de um elemento escolhido aleatoriamente ser rotulado incorretamente, e a "Entropia" que mede a desordem de um grupo.

Finalmente, a árvore de decisão foi instanciada com uma profundidade de 2. Em seguida, ocorreu o treinamento do modelo, a previsão dos valores de teste e o cálculo da pontuação, onde:

- **precision:** Proporção de verdadeiros positivos em relação ao total de instâncias classificadas como positivas pelo modelo.Mede a precisão do modelo quando ele prevê positivos.
- **recall:** Proporção de verdadeiros positivos em relação ao total de instâncias que realmente são positivas. Mede a capacidade do modelo de capturar todos os casos positivos.
- **accuracy:** proporção de todas as predições corretas em relação ao total de instâncias.Mede a precisão global do modelo.
- **f1-score:** Média harmônica entre precision e recall.

	precision	recall	f1-score	support
0	0.70	0.78	0.74	809
1	0.75	0.66	0.70	808
accuracy			0.72	1617

E com a matriz de confusão (Figura 6) podemos enxergar um valor razoável para o número de amostras que são ruins e foram classificadas como boas, o que é o pior caso para o nosso dataset.

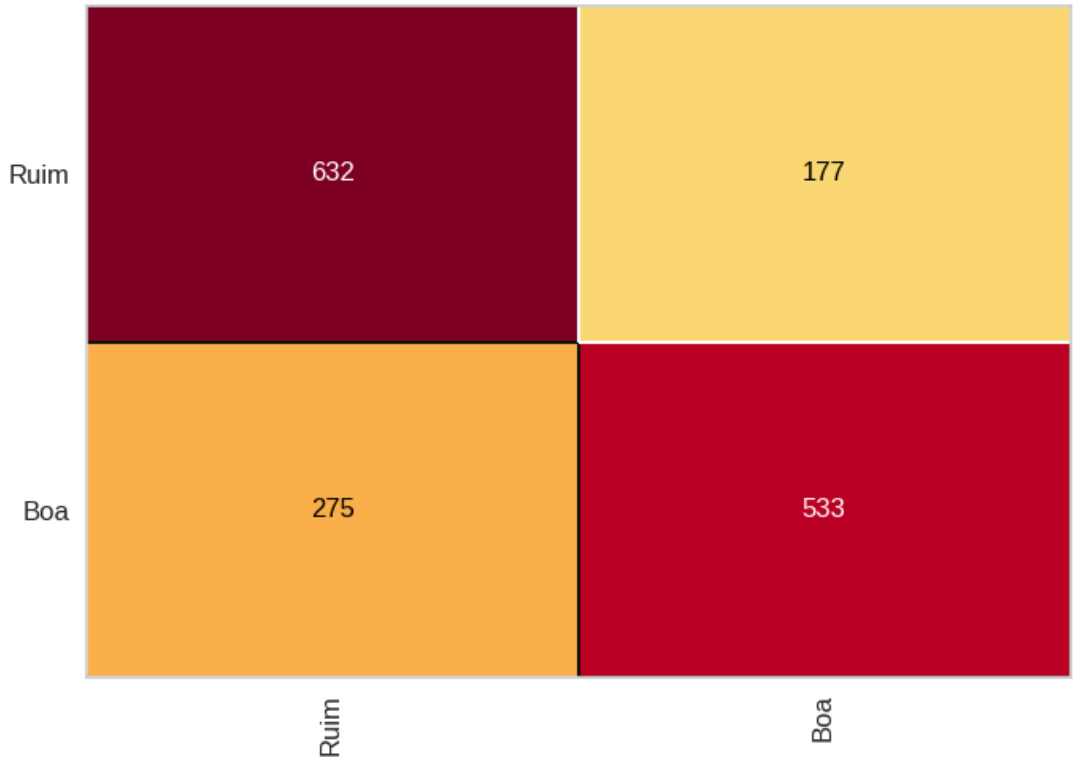

Figura 6

3.2 Naive Bayes:

O principal trunfo deste classificador é que mesmo que os atributos estejam inter-relacionados (o que não é o caso), eles são tratados como independentes para simplificar o cálculo das probabilidades. Houve a possibilidade de se utilizar um hiperparâmetro no modelo, o ***var-smoothing***, porém como o nível de complexidade foi relativamente alto comparado ao resultado obtido pela sua utilização, foi preferido não utilizar hiperparâmetros. Com este algoritmo possuímos os seguintes resultados:

	precision	recall	f1-score	support
0	0.69	0.60	0.64	809
1	0.65	0.73	0.69	808
accuracy			0.67	1617

Onde é possível enxergar uma acurácia consideravelmente baixa, em relação aos algoritmos testados aqui neste projeto. O destaque negativo acontece com a visualização da matriz de confusão (Figura 7), onde podemos ver um alto número de amostras ruins sendo classificadas como boas.

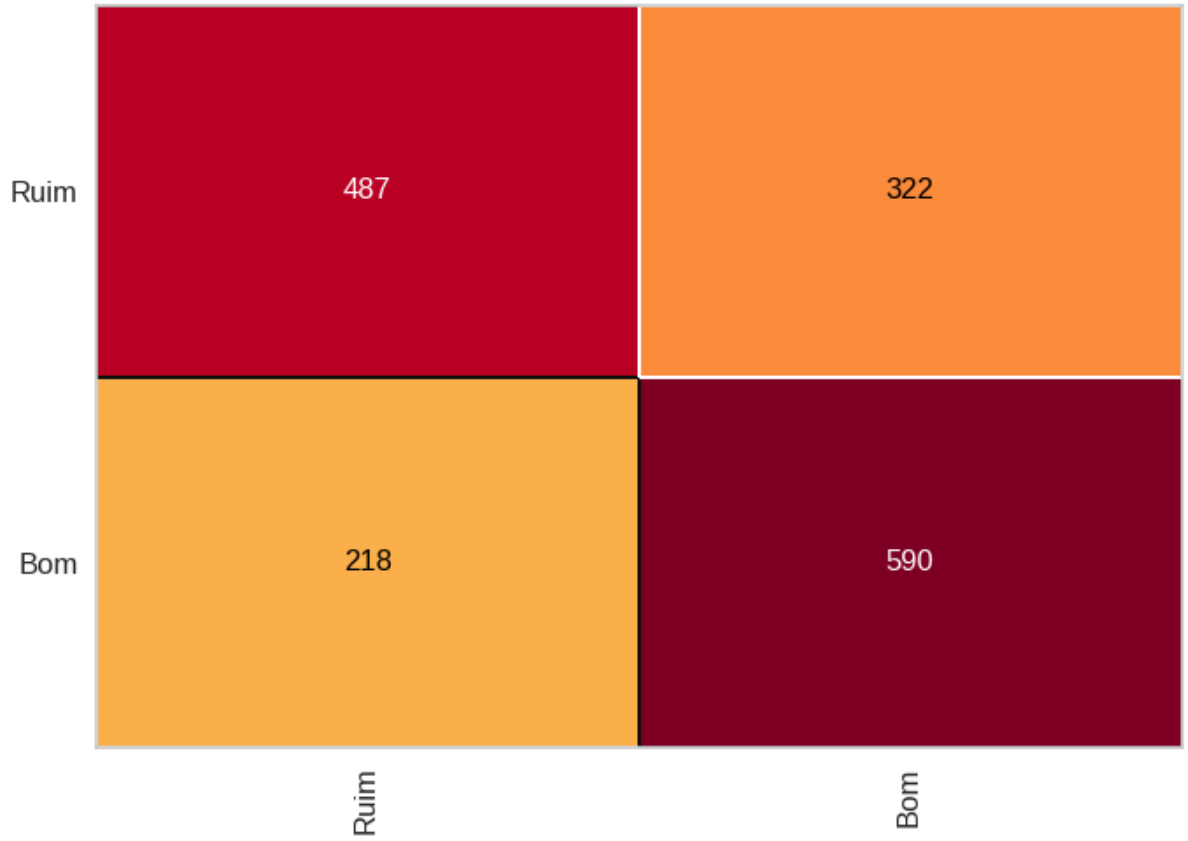

Figura 7

3.3 Regressão Logística:

Este é um modelo estatístico utilizado para relações binárias de variáveis independentes. Estima a probabilidade da ocorrência de um evento, limitada a 0 e 1. Nele tivemos os seguintes resultados:

	precision	recall	f1-score	support
0	0.67	0.75	0.71	809
1	0.71	0.63	0.67	808
accuracy			0.69	1617

Que são valores razoáveis em relação aos demais algoritmos, tendo como matriz de confusão (Figura 8) um resultado mais satisfatório em relação ao Naive Bayes.

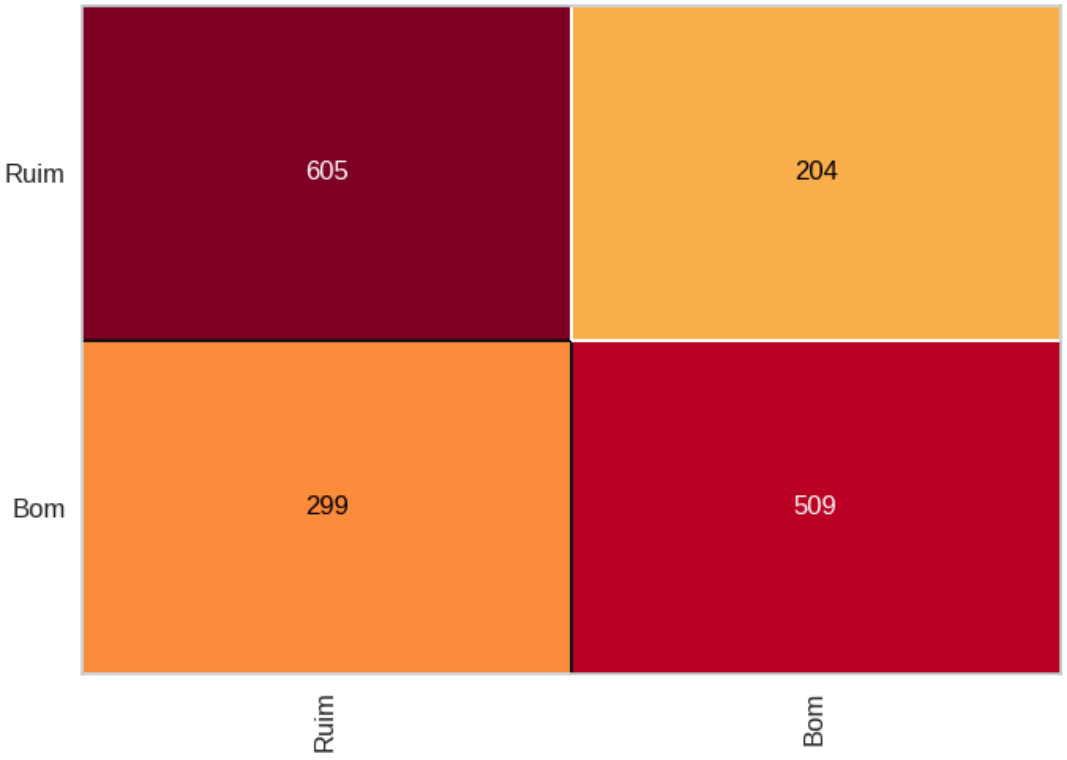

Figura 8

3.4 K-Vizinhos:

Para a obtenção do melhor K foi utilizado uma validação Cruzada, onde podemos enxergar que com 1 vizinho obtemos um score de 75%, sendo a melhor opção de escolha (Como é possível enxergar no Gráfico 1).

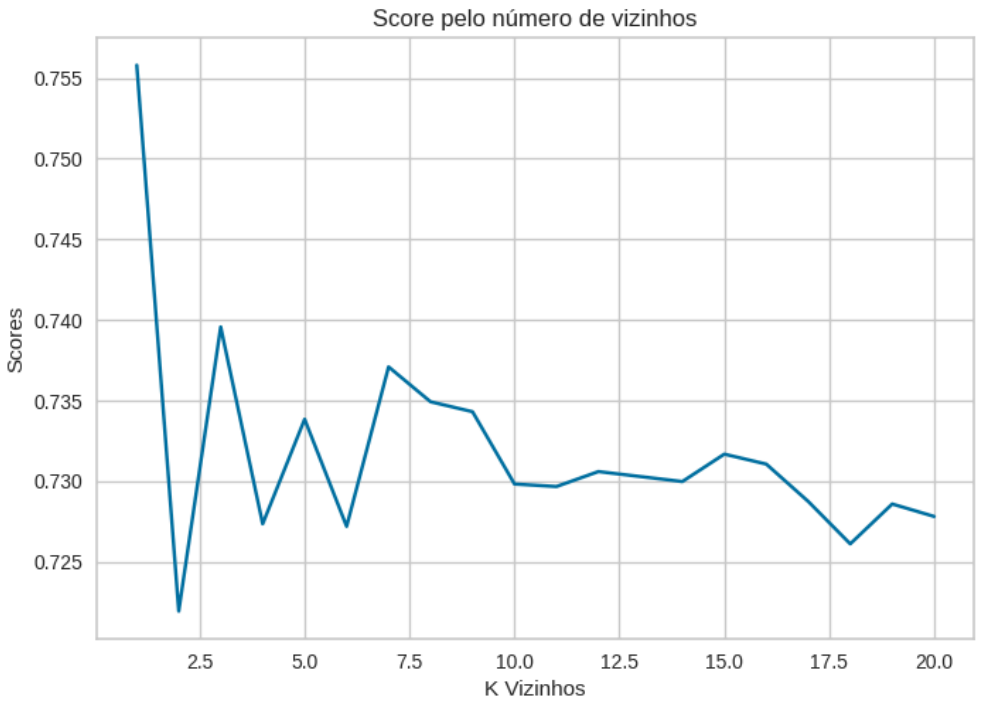

Gráfico 1

Abaixo estão os resultados obtidos utilizando K=1 e métrica da distância euclidiana temos:

	precision	recall	f1-score	support
0	0.79	0.85	0.82	809
1	0.84	0.77	0.80	808
accuracy			0.81	1617

Ademais ao utilizar a técnica PCA para diminuição de features para 2, podemos plotar o gráfico do KNN em duas dimensões resultando em:

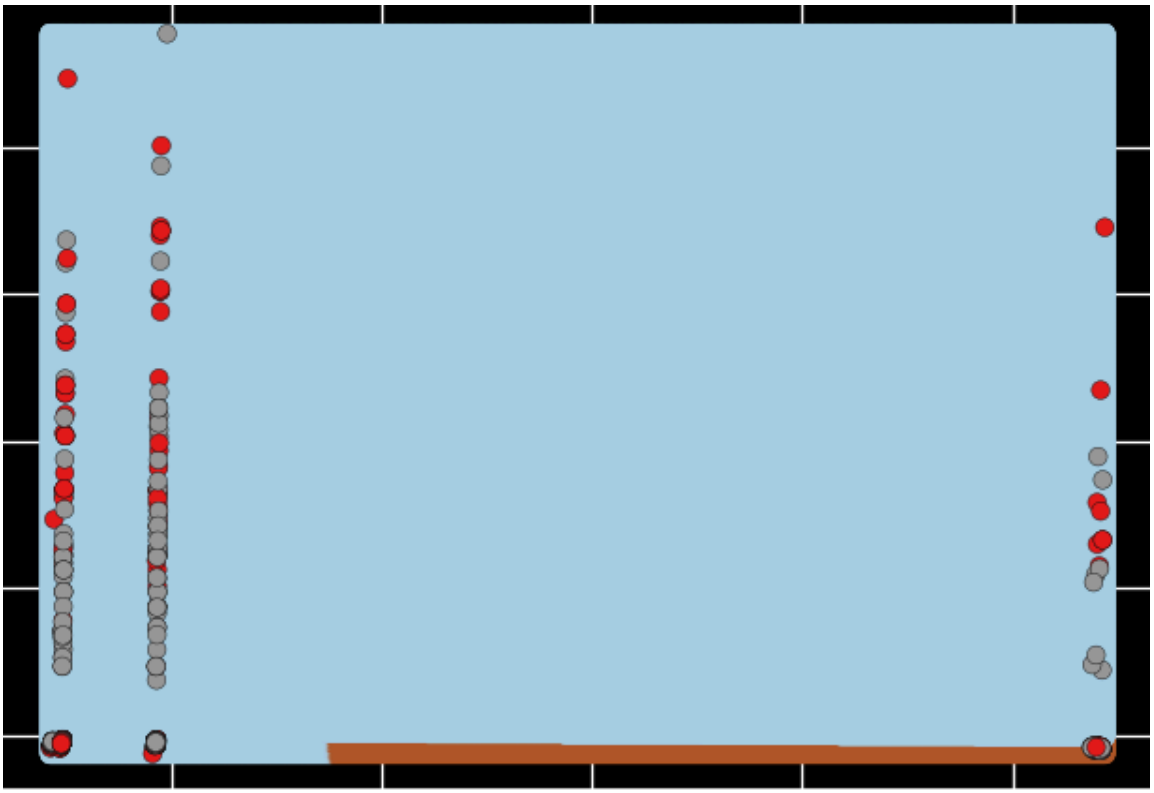

Gráfico 2

4. Resultados

Dos resultados finais obtidos, tivemos os seguintes.

DecisionTree: 74% de acurácia com desvio padrão de 6%
Naive Bayes: 65% de acurácia com desvio padrão de 4%
Logistic regression: 70% de acurácia com desvio padrão de 6%
K Neighbors: 76% de acurácia com desvio padrão de 7%

O K Neighbors Classifier apresentou desempenho superior em comparação com os outros modelos, destacando-se nas métricas de precision, recall e f1-score. A árvore de decisão e a regressão logística também demonstraram desempenhos competitivos, embora com uma variação moderada nos resultados. Já o Naive Bayes apresentou uma

acurácia mais baixa, porém com menor variação entre as diferentes validações.

O projeto ressaltou a importância do pré-processamento de dados, da escolha adequada de algoritmos e do ajuste de hiperparâmetros para alcançar resultados mais precisos na classificação da qualidade do vinho.