**Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas IFSULDEMINAS – Campus Poços de Caldas**

**Trabalho Prático de Inteligência Artificial – Análise Preditiva da Qualidade de Vinhos**

**Aluno:** Alessandro Augusto  
**Professor:** Douglas Castilho  
**Julho de 2025**

Este relatório detalha o processo de análise, pré-processamento e modelagem preditiva realizado sobre a base de dados de qualidade de vinhos, conforme as diretrizes do Trabalho Prático da disciplina de Inteligência Artificial.

### Parte 1: Análise e Pré-Processamento de Dados

#### **1. Identificação do Atributo Alvo (Saída)**

* **Atributo Original:** O atributo alvo original presente na base de dados é a coluna quality, uma variável numérica discreta que representa uma pontuação de 3 a 9 para cada vinho.
* **Transformação Aplicada:** Para criar um problema de classificação com maior aplicabilidade prática e para lidar com o desbalanceamento inerente às notas, a variável quality foi discretizada em três categorias ordinais.
* **Atributo Alvo Final:** O atributo alvo final, utilizado para treinar e avaliar os modelos, é a coluna quality\_category, com as seguintes classes nominais: **ruim** (notas de 3 a 4), **normal** (notas de 5 a 6) e **bom** (notas de 7 a 9).
* **Justificativa no Código:** A lógica para esta transformação está implementada na função processar\_dados\_para\_modelagem() do ficheiro cli.py e na função \_preprocess\_data() do ficheiro engine.py, onde o método pd.cut é utilizado para criar as categorias.

#### **2. Identificação dos Tipos de Dados dos Atributos de Entrada**

* **Atributo Qualitativo:** O projeto identifica corretamente um único atributo de entrada como qualitativo (ou categórico): color (cujos valores são 'red' ou 'white').
* **Atributos Quantitativos:** Os 11 atributos restantes são corretamente identificados como quantitativos (numéricos), pois representam medições físico-químicas: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol.
* **Justificativa no Código:** O código confirma esta distinção na etapa de pré-processamento. A função pd.get\_dummies é aplicada exclusivamente à coluna color para converter seus valores textuais ('red'/'white') num formato numérico (0/1). Este passo é característico do tratamento de dados qualitativos nominais, enquanto os atributos quantitativos, por já serem numéricos, são processados diretamente por outras técnicas como a padronização.

#### **3. Identificação da Escala de Dados dos Atributos de Entrada**

* **Escala Nominal:** O atributo qualitativo color está nesta escala. Seus valores ('red' e 'white') são rótulos que nomeiam categorias sem uma ordem ou hierarquia intrínseca.
* **Escala Intervalar:** O atributo quantitativo pH pertence a esta escala. É uma medida logarítmica onde os intervalos entre os valores são consistentes (a diferença entre pH 3 e 4 é a mesma que entre pH 5 e 6), mas não existe um ponto zero absoluto que represente a "ausência de acidez".
* **Escala Racional:** Os outros 10 atributos quantitativos estão nesta escala. Todos eles (fixed acidity, residual sugar, alcohol, etc.) possuem um zero absoluto e significativo (um valor 0 significa a ausência total daquela substância). Portanto, as razões entre os valores são válidas (ex: 10 g/L de açúcar é o dobro de 5 g/L).

#### **4. Exploração dos Dados Através de Medidas de Localidade**

A análise de medidas de localidade, como média e mediana (quartil 50%), foi realizada pela função analise\_exploratoria no cli.py. A execução desta função (opção 1 no menu CLI) gerou a seguinte tabela de estatísticas descritivas, que resume a tendência central de cada atributo:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Medidas de localidade e espalhamento (Itens 4 a 5)** | | | | | | | | |
|  | **Count** | **Mean** | **Std** | **Min** | **25%** | **50%** | **75%** | **Max** |
| **Fixed acidity** | 6497.0 | 7.215307 | 1.296434 | 3.80000 | 6.40000 | 7.00000 | 7.70000 | 15.90000 |
| **Volatile acidity** | 6497.0 | 0.339666 | 0.164636 | 0.08000 | 0.23000 | 0.29000 | 0.40000 | 1.58000 |
| **Citric acid** | 6497.0 | 0.318633 | 0.145318 | 0.00000 | 0.25000 | 0.31000 | 0.39000 | 1.66000 |
| **Residual sugar** | 6497.0 | 5.443235 | 4.757804 | 0.60000 | 1.80000 | 3.00000 | 8.10000 | 65.80000 |
| **Chlorides** | 6497.0 | 0.056034 | 0.035034 | 0.00900 | 0.03800 | 0.04700 | 0.06500 | 0.61100 |
| **Free súlfur dioxide** | 6497.0 | 30.525319 | 17.749400 | 1.00000 | 17.00000 | 29.00000 | 41.00000 | 289.00000 |
| **Total súlfur dioxide** | 6497.0 | 115.744574 | 56.521855 | 6.00000 | 77.00000 | 118.00000 | 156.00000 | 440.00000 |
| **Density** | 6497.0 | 0.994697 | 0.002999 | 0.98711 | 0.99234 | 0.99489 | 0.99699 | 1.03898 |
| **pH** | 6497.0 | 3.218501 | 0.160787 | 2.72000 | 3.11000 | 3.21000 | 3.32000 | 4.01000 |
| **Sulfates** | 6497.0 | 0.531268 | 0.148806 | 0.22000 | 0.43000 | 0.51000 | 0.60000 | 2.00000 |
| **Alchool** | 6497.0 | 10.491801 | 1.192712 | 8.00000 | 9.50000 | 10.30000 | 11.30000 | 14.90000 |
| **Quality** | 6497.0 | 5.818378 | 0.873255 | 3.00000 | 5.00000 | 6.00000 | 6.00000 | 9.00000 |

Observa-se, por exemplo, que a nota de qualidade (quality) média é de 5.82, enquanto a mediana é 6.0, indicando uma leve assimetria na distribuição das notas.

#### **5. Exploração dos Dados Através de Medidas de Espalhamento**

A mesma tabela acima, gerada pela função analise\_exploratoria, também fornece as medidas de espalhamento: desvio padrão (std), valor mínimo (min), valor máximo (max) e os quartis (25%, 50%, 75%), que compõem o intervalo interquartil. Essas medidas mostram a variabilidade dos dados. Por exemplo, o atributo total sulfur dioxide apresenta um desvio padrão alto (56.52) e uma grande amplitude (diferença entre max e min), sugerindo uma alta dispersão dos seus valores.

#### **6. Exploração dos Dados Através de Medidas de Distribuição**

A distribuição dos dados foi explorada visualmente através da função visualizar\_distribuicoes (opção 2 no CLI), que gera histogramas para cada atributo numérico, segmentados por cor de vinho. A versão cli\_manual.py oferece uma alternativa em texto que também cumpre o objetivo, mostrando a frequência de valores em diferentes faixas (bins). Essa análise permite identificar a forma da distribuição (ex: simétrica, assimétrica) de cada atributo.

#### **7. Identificação e Separação do Conjunto de Teste**

O conjunto de teste foi separado na função processar\_dados\_para\_modelagem utilizando a função train\_test\_split da biblioteca scikit-learn.

* **Técnica:** Foi utilizado o método **Hold-Out**, com 20% dos dados sendo reservados para o conjunto de teste (test\_size=0.2).
* **Representatividade:** Para garantir que o conjunto de teste fosse representativo, foi aplicada a **estratificação**. Notavelmente, foi criada uma coluna auxiliar (stratify\_col) que combina a categoria de qualidade (quality\_category) e a cor do vinho (color\_white). Ao usar esta coluna para a estratificação, o código garante que a proporção de cada categoria de qualidade, tanto para vinhos tintos quanto para brancos, seja a mesma nos conjuntos de treino e teste. Esta é uma abordagem robusta que preserva as características da população completa.

#### **8. Identificação e Eliminação de Atributos Não Necessários**

No pré-processamento, os únicos atributos eliminados diretamente foram quality (o alvo original, substituído por quality\_category), quality\_category e stratify\_col (que cumpriram seus papéis e não são atributos de entrada para os modelos).

Para a análise de redundância, a **Matriz de Correlação** (opção 3 no CLI) foi gerada. Observando a matriz, nota-se uma correlação muito forte (0.72) entre free sulfur dioxide e total sulfur dioxide. Embora não tenham sido eliminados manualmente, essa alta correlação justifica a aplicação de uma técnica de redução de dimensionalidade como o PCA, que é oferecida como uma opção no sistema.

#### **9. Identificação e Eliminação de Exemplos Não Necessários**

A análise de exemplos não necessários focou na remoção de dados duplicados.

* **Identificação e Ação:** Na função processar\_dados\_para\_modelagem, o código df.drop\_duplicates(inplace=True) é executado.
* **Resultado:** A saída do terminal confirma a eficácia desta etapa: 1. Removendo 1177 linhas duplicadas.... Isso indica que uma quantidade significativa de exemplos redundantes foi corretamente identificada e removida, melhorando a qualidade do conjunto de dados para o treinamento.

#### **10. Análise e Aplicação de Técnicas de Amostragem de Dados**

A base de dados original possui 6497 exemplos, um volume considerado suficiente para o treinamento dos modelos propostos, não exigindo uma técnica de amostragem para reduzir o tamanho do dataset. No entanto, uma técnica de amostragem foi aplicada para resolver outro problema, o de desbalanceamento, conforme detalhado no próximo item.

#### **11. Identificação e Aplicação de Técnicas para Minimizar Problemas de Desbalanceamento**

Após a discretização do atributo alvo quality nas categorias 'ruim', 'normal' e 'bom', o dataset tornou-se desbalanceado. A classe 'normal' é majoritária, enquanto 'ruim' e 'bom' são minoritárias.

* **Análise:** O desbalanceamento pode levar os modelos a terem um bom desempenho na classe majoritária, mas falharem em prever corretamente as classes minoritárias, que são muitas vezes as de maior interesse.
* **Técnica Aplicada:** Para mitigar este problema, a técnica **SMOTE (Synthetic Minority Over-sampling Technique)** foi corretamente aplicada. Na função processar\_dados\_para\_modelagem, a linha X\_train\_balanced, y\_train\_balanced = smote.fit\_resample(X\_train\_final, y\_train) realiza o rebalanceamento.
* **Justificativa:** É crucial notar que o SMOTE foi aplicado **apenas no conjunto de treino**, o que é a prática correta para evitar o vazamento de dados (data leakage) e garantir que o conjunto de teste permaneça uma representação real e não vista do problema.

#### **12. Limpeza de Dados**

* **a. Identificação e Eliminação de Ruídos ou Outliers:** A função visualizar\_distribuicoes gera **boxplots** para cada atributo, que são uma ferramenta visual padrão para a identificação de outliers. Embora o código permita a visualização, ele não aplica uma regra automática para remoção, o que é uma decisão de projeto válida, pois nem todo outlier é um erro e pode representar uma característica importante do domínio.
* **b. Identificação e Eliminação de Dados Inconsistentes:** Nenhuma inconsistência óbvia foi encontrada nos dados. Os valores estão dentro dos intervalos esperados para as medições físico-químicas de vinhos.
* **c. Identificação e Eliminação de Dados Redundantes:** Conforme detalhado no item 9, **1177 linhas duplicadas** foram identificadas e removidas.
* **d. Identificação e Resolução de Dados Incompletos (Ausentes):** A análise inicial na função analise\_exploratoria (saída do terminal) mostrou que df.isnull().sum() retorna 0 para todas as colunas. Portanto, **não havia dados ausentes**, e nenhuma técnica de preenchimento foi necessária.

#### **13. Identificação e Conversão dos Tipos de Dados**

* **a. Conversão de Tipos:**

**Nominal para Binário:** O atributo color (categórico nominal) foi convertido para um formato numérico binário (0 para 'red', 1 para 'white') usando pd.get\_dummies.

**Numérico para Ordinal:** O atributo quality (numérico) foi convertido para quality\_category (categórico ordinal: ruim < normal < bom) usando pd.cut.

* **b. Normalização dos Dados:** Foi utilizada a **Padronização (Standardization)** através do StandardScaler do Scikit-learn. Esta técnica transforma os dados para que tenham média 0 e desvio padrão 1. A padronização é essencial para algoritmos sensíveis à escala das features, como o K-NN (que se baseia em distâncias) e Redes Neurais (para uma convergência mais rápida e estável do gradiente).

#### **14. Análise e Aplicação de Técnica para Redução de Dimensionalidade**

O projeto investigou a redução de dimensionalidade como uma etapa experimental.

* **Técnica Aplicada:** Foi utilizada a **Análise de Componentes Principais (PCA)**. O PCA é uma técnica de extração de características que transforma o conjunto original de atributos correlacionados em um novo conjunto de atributos não correlacionados (componentes principais), ordenados pela quantidade de variância que explicam.
* **Implementação:** A opção 9 do menu CLI ativa o pipeline com PCA. A linha pca = PCA(n\_components=0.95) no código indica que o objetivo é manter **95% da variância original** dos dados.
* **Resultado:** A saída do terminal informa: PCA selecionou 9 componentes dos 12 originais. Isso significa que foi possível reduzir a dimensionalidade do problema em 25% (de 12 para 9 atributos) perdendo apenas 5% da informação de variância, o que pode levar a modelos mais simples e rápidos sem uma perda significativa de performance.

### Parte 2: Análise Preditiva

#### **1. Definição da Técnica de Validação**

O projeto utilizou duas técnicas de validação de forma apropriada:

1. **Hold-Out:** Na avaliação principal dos modelos (opções 4 a 9), foi usada uma única divisão dos dados em 80% para treino e 20% para teste. Esta abordagem é rápida e eficaz para uma avaliação inicial.
2. **Validação Cruzada (Cross-Validation):** Na opção 10, foi utilizada a **Validação Cruzada Estratificada com 5 Folds (StratifiedKFold)**. Este método divide o conjunto de treino em 5 partes, usando 4 para treinar e 1 para validar, repetindo o processo 5 vezes. O resultado é a média e o desvio padrão da performance, fornecendo uma estimativa muito mais robusta e confiável da capacidade de generalização do modelo e de sua estabilidade.

#### **2. Definição das Métricas de Avaliação**

O projeto utilizou um conjunto abrangente de métricas para uma avaliação completa:

* **Matriz de Confusão:** Apresentada para todos os modelos, detalhando os acertos e erros por classe.
* **Acurácia:** Percentual geral de acertos.
* **Precisão:** Dos que foram classificados como uma classe, quantos realmente eram.
* **Recall (Sensibilidade):** De todos que pertenciam a uma classe, quantos foram corretamente classificados.
* **F1-Score:** Média harmônica entre precisão e recall, útil para dados desbalanceados.
* **Especificidade:** Média da capacidade do modelo de identificar corretamente os verdadeiros negativos. Foi implementada uma função calcular\_especificidade\_media para esta métrica.

A apresentação das métricas de forma ponderada (weighted avg) e por tipo de vinho (tinto vs. branco) enriquece enormemente a análise.

#### **3. Definição e Análise do Algoritmo Baseline**

* **Definição:** O algoritmo DummyClassifier com a estratégia "most\_frequent" foi usado como baseline. Este modelo simplesmente classifica todos os exemplos como pertencentes à classe mais frequente nos dados de treino.
* **Análise (Pipeline Padrão):** O baseline obteve uma **acurácia de 0.1898**. Como ele prevê apenas a classe "bom" (que era a mais frequente *após o SMOTE* no treino), sua precisão e recall para as classes "normal" e "ruim" são 0. Este resultado é fundamental, pois estabelece o limiar mínimo de performance: qualquer modelo útil deve superar significativamente este valor.

#### **4-6. Criação de Modelos Preditivos**

Foram criados e avaliados três modelos de aprendizado de máquina, com hiperparâmetros pré-definidos:

* **K-NN (K-Nearest Neighbors):** n\_neighbors=7, weights='distance', metric='manhattan'.
* **Árvore de Decisão:** max\_depth=8, min\_samples\_split=5, criterion='gini'.
* **Rede Neural (MLP):** Duas camadas ocultas com 50 e 30 neurônios, ativação 'relu', otimizador 'adam'.

#### **7-8. Análise dos Resultados**

A análise comparativa dos resultados (obtidos pela opção 8 do CLI) permite extrair as seguintes conclusões:

**Pipeline Padrão (Sem PCA):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Modelo** | **Acurácia geral** | **F1-score (ponderado)** | **Acurácia (brancos)** | **Acurácia (tintos)** |
| **Baseline** | 0.1898 | 0.06 | 0.2083 | 0.1360 |
| **K-NN** | 0.6109 | 0.65 | 0.6048 | 0.6287 |
| **Árvore de decisão** | 0.5451 | 0.59 | 0.5253 | 0.6029 |
| **Rede neural (MLP)** | 0.6898 | 0.71 | 0.6705 | 0.7463 |

* **Melhor Modelo:** A **Rede Neural (MLP)** foi o modelo com o melhor desempenho geral, alcançando a maior acurácia (69%) e F1-Score (71%).
* **Desempenho por Cor:** Interessantemente, todos os modelos tiveram um desempenho notavelmente superior na classificação de **vinhos tintos** em comparação com os brancos. A MLP, por exemplo, alcançou 74.6% de acurácia nos vinhos tintos. Isso pode sugerir que os atributos físico-químicos são melhores preditores da qualidade em vinhos tintos.
* **Análise das Classes:** Todos os modelos tiveram dificuldade em classificar a classe "ruim", que possui o menor número de amostras, mesmo após o SMOTE. O recall para esta classe foi baixo em todos os cenários, indicando que os modelos ainda tendem a confundi-la com as outras.

**Pipeline com PCA:**

|  |  |
| --- | --- |
| **Modelo** | **Acurácia geral (com PCA)** |
| **Baseline** | 0.1898 |
| **K-NN** | 0.6006 |
| **Árvore de decisão** | 0.5846 |
| **Rede neural (MLP)** | 0.6917 |

* **Impacto do PCA:** A aplicação do PCA teve um impacto misto. Para o K-NN, a performance caiu ligeiramente. Para a Árvore de Decisão, houve uma melhora. Para a MLP, a performance se manteve praticamente a mesma (uma leve melhora de ~0.2%). O resultado mais importante aqui é que a MLP conseguiu manter (e até melhorar um pouco) seu desempenho superior utilizando 3 atributos a menos, o que a torna um modelo mais eficiente.

**Análise de Estabilidade (Validação Cruzada):**

|  |  |  |
| --- | --- | --- |
| **Modelo** | **Acurácia média (CV)** | **Desvio padrão (CV)** |
| **K-NN** | 0.7834 | ± 0.0065 |
| **Árvore de decisão** | 0.7580 | ± 0.0120 |
| **Rede neural (MLP)** | 0.7834 | ± 0.0089 |

* **Acurácia no Treino vs. Teste:** A validação cruzada, realizada nos dados de treino (antes do SMOTE), mostra acurácias mais altas (em torno de 78%) do que as obtidas no conjunto de teste final (~69% para MLP). Isso é esperado e evidencia a importância do conjunto de teste para avaliar a generalização real.
* **Estabilidade:** O K-NN e a MLP apresentaram não apenas as maiores acurácias médias, mas também os menores desvios padrão, indicando que são modelos mais estáveis e consistentes em diferentes subconjuntos dos dados. A Árvore de Decisão, com um desvio padrão maior, mostrou-se um pouco mais instável.

### Conclusão Final do Projeto

O projeto foi executado com sucesso, seguindo um pipeline de pré-processamento e análise de dados completo e bem fundamentado. As etapas de limpeza, transformação e balanceamento foram cruciais para a obtenção de resultados significativos.

Dentre os modelos avaliados, a **Rede Neural (MLP)** demonstrou ser o algoritmo mais eficaz para a tarefa de classificar a qualidade dos vinhos, tanto no pipeline padrão quanto no pipeline com redução de dimensionalidade via PCA. Os resultados da validação cruzada também reforçam a robustez do modelo MLP. A análise demonstrou que, embora a predição da qualidade do vinho seja uma tarefa complexa, os modelos de aprendizado de máquina conseguem um desempenho consideravelmente superior a uma classificação aleatória ou de base, com destaque para a performance na distinção da qualidade de vinhos tintos.