UCI Open Data: Previsão do teor alcoólico presente em vinhos

1º João Lucas Oliveira Mota
Departamento de Engenharia de Teleinformática
Universidade Federal do Ceará
Fortaleza, Brasil
jlucasoliveira2002@alu.ufc.br

2º João Lucas Lima Monteiro
Departamento de Engenharia de Teleinformática
Universidade Federal do Ceará
Fortaleza, Brasil
joaolucaslima@alu.ufc.br

***Abstract*— Nesse artigo, utilizamos o dataset Wine Quality com medições físico-químicas de diferentes amostras de vinhos. Entre essas medições, destaca-se o teor alcoólico, um fator crucial na avaliação da qualidade e sabor dos vinhos. Neste artigo, realizamos uma análise abrangente dos modelos de regressão linear e suas variantes, como Ridge Regression, a fim de prever o teor alcoólico dos vinhos com base nas demais características do dataset. Ademais, foi realizada a análise desses mesmos dados segundo um modelo de redes neurais para uma análise de possíveis relações não-lineares. Além dos métodos tradicionais, também empregamos abordagens de aprendizado profundo com redes neurais para prever o teor alcoólico dos vinhos. Comparou-se os resultados dessas técnicas com os modelos de regressão clássicos, identificando os métodos que proporcionam previsões mais precisas.**

Keywords—Vinhos, análise estatística, regressão linear, redes neurais.

I. Introdução .

No *dataset Wine Quality*[1], foram realizadas medições de diferentes características físico-químicas e sensoriais de diferentes vinhos. Entre essas medições encontra-se o teor alcóolico desses vinhos.

Neste artigo, realizamos uma análise de diferentes modelos de regressão linear, além de uma análise por métodos de redes neurais, buscando definir modelos preditivos para o teor alcoólico dos vinhos do *dataset*

II. Métodos

Nesta seção, vamos abordar os dados que estamos trabalhando, explicar os métodos utilizados e a teoria por trás deles

A. Conhecimento necessário

Focamos, neste trabalho, em análises de regressões lineares com e sem penalização, além de utilizarmos a técnica da PCR. Ademais, realizamos uma análise com base em redes neurais, especificamente com base no modelo Multilayer Perceptron.

1) Regressão

Nesse artigo, realizamos os processos de regressão linear simples e penalizada, além do processo de PCR.

Para a regressão linear simples, visamos encontrar um modelo linear que melhor se adeque as relações entre as variáveis do dataset, permitindo fazer inferências com base nesses dados.

Para tanto, utilizamos a fórmula 1, com "Y" como a variável dependente, "X_i" as variáveis independentes, "β" os coeficientes de regressão e "ε" o chamado erro aleatório.

$$Y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \varepsilon \qquad (1)$$

Em seguida, realizamos o procedimento de regressão penalizada, mais especificamente segundo o método de Regressão de Ridge, que busca a diminuição de sobreajuste ao adicionar o fator de penalização L2, especificado na fórmula 2 abaixo.

$$L2 = \alpha \sum_{i=1}^{p} \beta_i^2 \qquad (2)$$

Nessa fórmula, "β" segue representando os coeficientes e "α" é o hiperparâmetro de penalização, que controla o grau de regularização aplicado.

Além disso, realizamos o processo de Regressão por Componente Principal (PCR), técnica que combina a Análise de Componentes Principais com uma regressão linear múltipla.

Tal técnica é utilizada para abordar problemas de regressão em conjuntos de dados de alta dimensionalidade, onde as variáveis independentes estão correlacionadas, como é o caso do dataset utilizado.

Há ainda, 2 valores relevantes que as técnicas de regressão nos entregam: o Root Mean Square Error (RMSE), e o coeficiente de determinação. Segue as equações que os representam.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{X}_i)} \qquad (3)$$

Na equação 3, o valor de Y_i representa o valor real (ou observado) para a i-ésima amostra, o $\hat{Y}_i$ representa a previsão do modelo para a i-ésima amostra, e n é o número total de amostras.

$$R^2 = 1 - \frac{SSR}{SST} \qquad (4)$$

Para a equação 4, o valor de SSR (Sum of Squares of Residuals) é a soma dos quadrados dos resíduos, que mede a variação não explicada pelo modelo, enquanto SST (Total Sum of Squares) é a soma total dos quadrados, que mede a variação total nos dados

2) Redes Neurais

Por fim, realizamos uma análise dos dados segundo o modelo *Multi-Layer Perceptrons,* que simula neurônios artificiais para a identificação de padrões não lineares, através de conexões ponderadas nas camadas ocultas, que têm como objetivo a identificação de padrões não lineares nos dados inseridos.

Utilizamos a biblioteca "*scikit-learn"*, com a classe "*MLPRegressor"* e suas funções associadas para implementar tal modelo no código.

B. Pré-processamento

A priori, ao checarmos a matriz de correlação dos preditores, reparamos um dos preditores com uma baixíssima correlação com a coluna target escolhida (álcool). Portanto, tal coluna, a de Sulfatos, foi retirada do dataset para facilitar as análises posteriores.

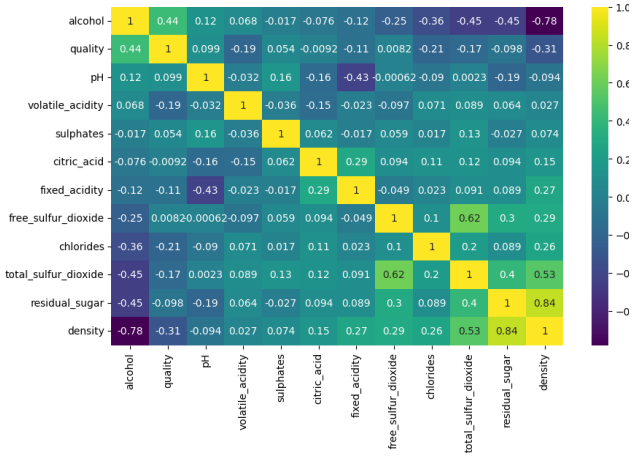


Figura 1: Matriz de correlação dos preditores

Em seguida, verificamos que os dados possuem muitos *outliers*, conforme a Figura 1 ilustra. Portanto, realizamos o procedimento remoção de *outliers*, visando eliminar tais pontos.

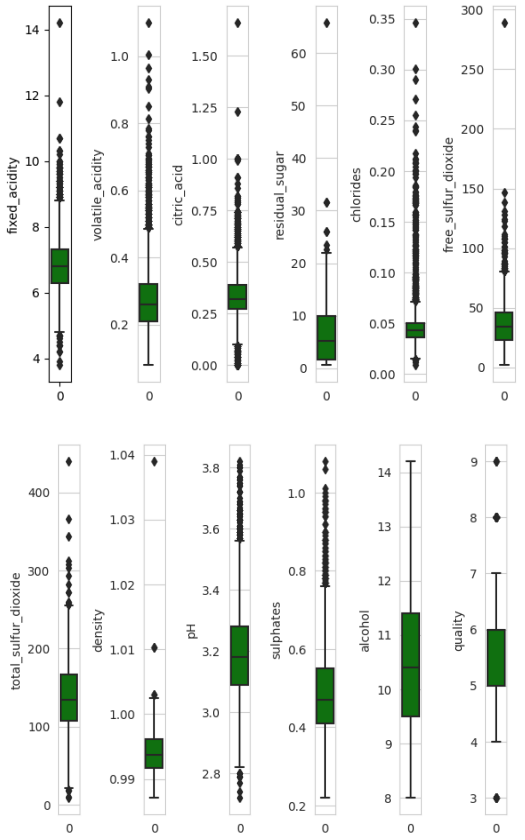


Figura 2: Box-plot dos preditores antes do pré-processamento

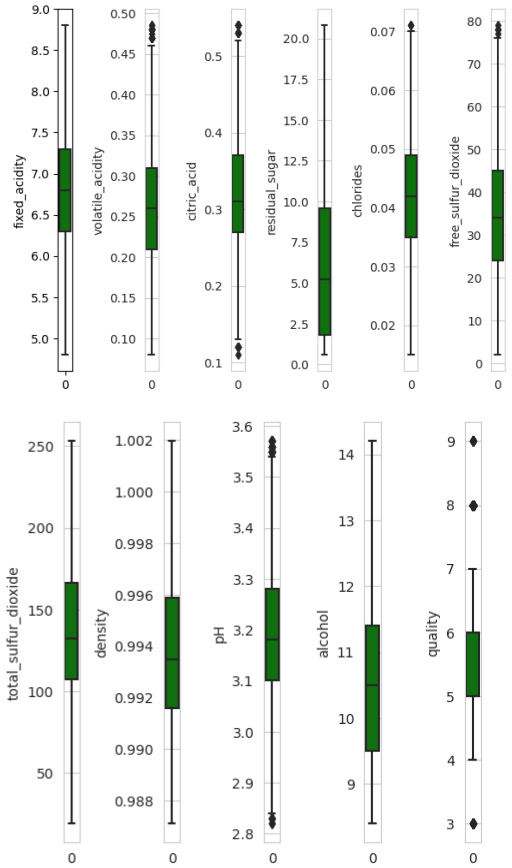


Figura 3: Box-plot dos preditores depois do pré-processamento

A Figura 3 ilustra a diminuição dos *outliers* presentes no dataset, além de deixar claro a eliminação da coluna "sulfatos", devido a sua baixa correlação com a coluna alvo.

Além disso, realizamos o procedimento de checagem e diminuição da assimetria, em especial para os casos com a assimetria mais alta. Segue as tabelas dos valores de assimetria antes e depois desse processo. Dentre os valores de destaque, temos o de Acidez volátil, Cloretos e Dióxido de Enxofre livre, além do valor de Sulfatos, que foi retirado durante o pré-processamento.

Tabela 1: Valores de assimetria antes e depois do pré-processamento

Preditor	Assimetria Inicial	Assimetria Final
fixed_acidity	0.647751	0.164060
volatile_acidity	1.576980	-0.011320
citric_acid	1.281920	0.374180
residual_sugar	1.077094	0.737489
chlorides	5.023331	-0.019602
free_sulfur_dioxide	1.406745	-0.057751
total_sulfur_dioxide	0.390710	0.300178
density	0.977773	0.304256
pH	0.457783	0.195610
sulphates	0.977194	Valor retirado
alcohol	0.487342	0.386442
quality	0.155796	0.169765

Por fim, realizamos o processo de *one-hot encoding,* visando a criação de variáveis binárias, através do método *get_dummies()* da biblioteca *Pandas.* Dessa forma, transformamos cada valor do preditor *quality* em uma coluna e após isso preenchendo com valor 0 para demonstrar a ausência do valor daquela coluna e 1 para presença do valor daquela coluna

III. Resultados

Neste tópico, apresentaremos os resultados das manipulações e técnicas apresentadas.

A. Modelo de Regressão Linear Implementado e validação cruzada

No modelo de Regressão Linear Ordinária, utilizamos duas funções, uma delas implementada e a outra importada da biblioteca *"scikit-learn"*, em Python.

Para a função implementada, o programa convergiu após 869 iterações, gerando um RMSE = 0.361 e um R^2 = 0.91569. Já para a função importada da biblioteca, os valores são 0.359 e 0.916, respectivamente RMSE e R^2, muito próximos do esperado e dos valores implementados.

Em seguida realizamos o processo de regressão com validação cruzada com 5-*fold* de forma implementada e utilizando funções da biblioteca, gerando os valores de RMSE e R^2 presentes nas Tabelas 2 e 3

Tabela 2: Regressão implementada com cross validation implementada

Fold	RMSE	R^2	Tempo
0	0.336651	0.922672	0.224015
1	0.379040	0.897429	0.214322
2	0.372850	0.910571	0.209761
3	0.348692	0.915259	0.208700
4	0.359355	0.910502	0.198805

Tabela 3: Regressão da biblioteca com cross validation da biblioteca

Fold	RMSE	R^2	Tempo
0	-0.328647	0.926305	0.022284
1	-0.371510	0.901464	0.008173
2	-0.368789	0.912509	0.018380
3	-0.342249	0.918361	0.015164
4	-0.359409	0.910475	0.017419

Dentre os valores das Tabelas 2 e 3, podemos destacar uma semelhança muito grande entre os RMSE e R^2 para cada iteração (quando considerados os módulos dos valores, devido a uma convenção da regressão da biblioteca que entrega os valores de RMSE negativos). Também é possível destacar uma grande discrepância no tempo de execução, com as funções da biblioteca demandando um tempo de execução 10 vezes menor que as implementadas.

B. Modelo de Regressão Linear penalizado por L2

No modelo de Regressão Linear Penalizada, especificamente para o modelo de regressão Ridge, utilizamos novamente funções implementadas e posteriormente importadas da biblioteca *"scikit-learn"*, em Python.

Para a função implementada, o programa convergiu após 1263 iterações utilizando os dados de teste, gerando um RMSE = 0.358 e um R2 = 0.917007, enquanto para a função importada da biblioteca, os valores são 0.35920 e 0.916753 (respectivamente RMSE e R^2) com esse mesmo conjunto de dados. Novamente, os valores implementados e importados resultam em RMSE e R^2 muito próximos do esperado e entre si.

Em seguida, conduzimos o processo de regressão Ridge com validação cruzada de 5-*fold*, tanto através de

implementação própria quanto fazendo uso de funções da biblioteca, resultando nos valores de RMSE e R^2 registrados nas Tabelas 4 e 5.

Tabela 4: Regressão de Ridge Implementada com Cross validation implementada.

Fold	RMSE	R^2	Tempo
0	0.329843	0.925768	0.464993
1	0.372838	0.900758	0.422524
2	0.369000	0.912408	0.583775
3	0.342747	0.918124	0.385902
4	0.358109	0.911122	0.634983

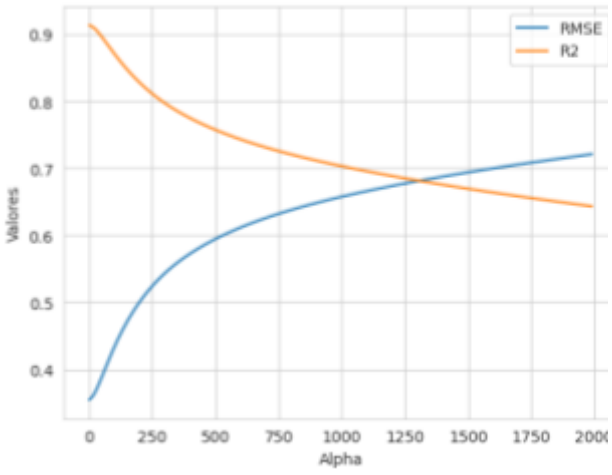
Tabela 5: Regressão de Ridge da biblioteca com cross validation da biblioteca

Fold	RMSE	R^2	Tempo
0	-0.328435	0.926400	0.012032
1	-0.371569	0.901433	0.011090
2	-0.368832	0.912488	0.014718
3	-0.341826	0.918563	0.003973
4	-0.359201	0.910579	0.016855

Novamente, os valores obtidos durante as duas implementações são muito próximos em módulo (atente-se novamente à convenção de sinais explicada anteriormente). Também se repete a discrepância da performance dos algoritmos implementados quando comparados aos da biblioteca, que demandam significativamente menos tempo.

Em seguida, calculamos o *alpha* ideal utilizando uma validação cruzada de 5-*fold*. O Figura 4 mostra os valores possíveis de *alpha* em função dos RMSE e R^2.

Figura 4: Representação dos valores de RMSE e R^2 em função de Alpha



A partir dos dados representados por esse gráfico, utilizamos de uma função para escolher o valor ótimo de *Alpha*, com base no que resulta no menor valor para o RMSE e o maior para o R^2, respectivamente 0.3545 e 0.9136. Para esses valores, a função retornou um valor $\alpha = 0.0001$. Tal valor condiz com o esperado, pois um menor alpha significa uma penalização menor.

Com esse valor calculado, refizemos os cálculos com as funções de Regressão Ridge anteriores, utilizando os valores de teste, gerando os valores apresentados nas tabelas 6 e 7.

Tabela 6: Regressão de Ridge Implementada com Cross validation implementada, aplicando o melhor α.

Fold	RMSE	R^2	Tempo
0	0.339266	0.924888	0.276427
1	0.382120	0.897713	0.413519
2	0.347860	0.921733	0.330345
3	0.374735	0.920704	0.257111
4	0.358088	0.912549	0.338759

Tabela 7: Regressão de Ridge da biblioteca com cross validation da biblioteca, aplicando o melhor α.

Fold	RMSE	R^2	Tempo
0	0.340240	0.924457	0.005711
1	0.381536	0.898025	0.007225
2	0.346746	0.922234	0.009104
3	0.372551	0.921626	0.002060
4	0.357988	0.912597	0.005778

Assim, é possível comparar os resultados antes e depois da aplicação de α, com os valores de RMSE e R^2 levemente mais próximos do ideal. Entretanto, tal diferença acentua-se nos valores de desempenho, reduzindo significativamente tanto para a função implementada quanto para a importada.

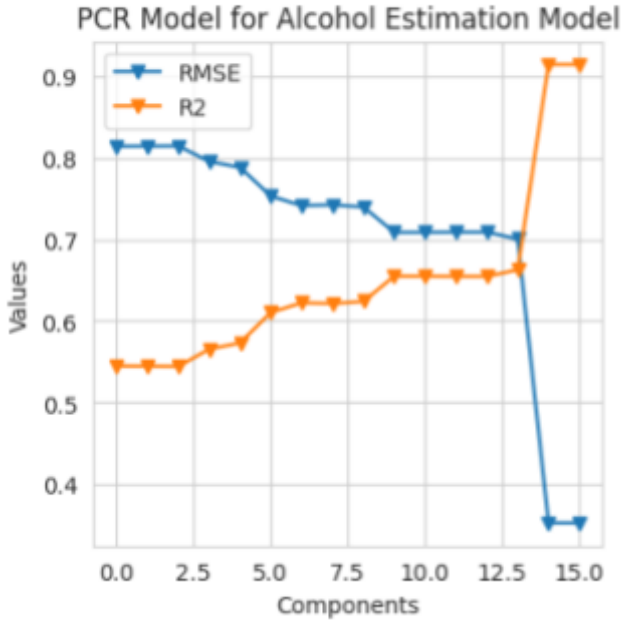
C. Regressão por Componente Principal

Para o modelo de Regressão por Componente Principal, o processo iniciou-se com com a aplicação da técnica de Análise de Componentes Principais, com o número de componentes = 16, ou seja, sem redução da dimensão do dataset.

Em seguida, para o processo de PCR, foram realizadas iterações da técnica de *cross-validation* (utilizando o conjunto de dados de treino), começando com um componente, adicionando-os um a um e verificando a progressão dos valores de RMSE e R^2 a cada iteração. Tal processo repetiu-se até a obtenção de valores satisfatórios para cada uma dessas variáveis.

O Figura 5 ilustra os resultados obtidos. Note que os valores ideais somente foram atingidos nas últimas iterações desse processo.

Figura 5: Representação dos valores de RMSE e R^2 em função do número de



Tal discrepância nos valores de RMSE e R^2 antes e depois do 14º componente indicam que o modelo fica melhor otimizado para quantidades de componente maiores que 14.

Quando aplicados os valores do conjunto de teste, o valor de "n" componentes que mais otimiza os valores buscados é n = 16, com de RMSE = 0.359415 e R^2 = 0.91665, enquanto para n = 14 os valores são RMSE = 0.67147 e R^2 = 0.7135, demonstrando que nesse caso a redução de dimensionalidade não ajuda no desempenho do algoritmo pois há perda de informação razoável.

D. Rede neural

Por fim, para a implementação de Redes Neurais do modelo *Multi-Layer Perceptrons*, realizamos a implementação com as funções presentes na biblioteca "*scikit-learn"*, conforme foi citado anteriormente.

Realizando o procedimento antes e depois da *cross-validation* com 10-*fold* (utilizando o conjunto de treino) temos os seguintes resultados de RMSE = 0.33652 e R^2 = 0.926934 para antes, com RMSE = 0.34561 e R^2 = 0.917894 resultando do procedimento pós *cross-validation*.

Finalmente, conduzimos a aplicação do modelo com os dados do conjunto de teste, o que resultou em uma precisão representada pelos seguintes valores: RMSE = 0.343909 e R^2 = 0.92369.

Dessa forma, observa-se que no caso de teste o modelo de rede neural se saiu com e R^2 levemente melhor que a mesma estatística do modelo de regressão linear e um pouco pior do que a regressão linear penalizada, com os valores de RMSE igualmente satisfatórios para ambos os casos.

Assim, conclui-se que o modelo de rede neural superou em parte os modelos lineares desenvolvidos anteriormente, e que isso se deve a uma mescla no conjunto de dados relacionado de forma linear e não linear. De certa forma, o conjunto de dados possui preditores muito relacionados de forma linear e outros não tão relacionados, mas que podem ser preditos com o auxílio das redes neurais.

IV. Referências

[1] 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. ISSN: 0167-9236. Disponível em: https://archive.ics.uci.edu/dataset/186/wine+quality

[2] JAMES, Gareth *et al*. **An Introduction to Statistical Learning with Applications in Python**. S.I: Springer, 2023. Disponível em: https://hastie.su.domains/ISLP/ISLP_website.pdf. Acesso em: 10 set. 2023.

[3] VARELLA, Carlos Alberto Alves. **Análise de Componentes Principais**. Disponível em: http://www.ufrrj.br/institutos/it/deng/varella/Downloads/multivariada%20aplicada%20as%20ciencias%20agrarias/Aulas/analise%20de%20componentes%20principais.pdf. Acesso em: 10 set. 2023