Análise Comparativa de Modelos de Regressão para a Predição da Solubilidade Aquosa de Compostos Químicos

1st José Ferreira Lessa
Dept. of Teleinformatics Engineering
Universidade Federal do Ceara
Fortaleza, Brazil
josefflessa@alu.ufc.br

2rd Nataniel Marques Viana Neto
Dept. of Teleinformatics Engineering
Universidade Federal do Ceara
Fortaleza, Brazil
natanielmarques@alu.ufc.br

3nd Matheus Rocha Gomes da Silva
Dept. of Teleinformatics Engineering
Universidade Federal do Ceara
Fortaleza, Brazil
theusrocha2004@alu.ufc.br

4th Victor Guedes Alves Texeira
Dept. of Teleinformatics Engineering
Universidade Federal do Ceara
Fortaleza, Brazil
victorguedes80@alu.ufc.br

***Abstract*—Este trabalho discorre da previsão da solubilidade aquosa de diversos compostos químicos, utilizando o conjunto de dados composto por 1267 observações e 228 preditores, incluindo fingerprints binários, descritores de contagem e variáveis contínuas. Foi feita a análise exploratória dos dados bem como o pré-processamento desses para reduzir assimetrias e padronizar as escalas. Em seguida foi feita a análise de modelos de regressão por mínimos quadrados ordinários (OLS), modelos penalizados (Ridge e Lasso), regressão por componentes (PCR e PLS) e redes neurais. Cada análise foi feita comparando implementações manuais e prontas do scikit-learn. Além disso, foi feita a avaliação dos modelos com métricas como o root mean squared error (RMSE) e coeficiente de determinação (R^2). Os resultados mostram que modelos regularizados e baseados em componentes superam o OLS tradicional, enquanto a rede neural apresenta um desempenho superior. Por fim, o estudo fornece uma base sólida para futuras implementações e refinamento do modelo preditivo.**

***Index Terms*—solubilidade, EDA, compostos químicos, regressão, componentes, métricas**

I. Introdução

A representação de compostos químicos pode ser feita por meio de estruturas moleculares e fórmulas químicas, que descrevem a organização espacial e atômica das moléculas. A partir dessas representações, é possível derivar medidas quantitativas conhecidas como descritores químicos, fundamentais para modelagem preditiva. Com isso, temos que esses descritores capturam as propriedades dessas moléculas, como números de carbonos e área superficial.

Mas nem todas propriedades relevantes de uma molécula pode ser determinada somente pela sua estrutura. A atividade biológica de um composto, por exemplo, geralmente depende de interações entre essa molécula e alvos biológicos como as proteínas. Entretanto, a sua estrutura química ainda sim, é usada para prever suas propriedades físico-químicas e a etapa que analisa isso é conhecida como modelagem de relação quantitativa estrutura-atividade (Quantitative Structure–Activity Relationship, QSAR)(Leach e Gillet, 2003).

Embora a atividade seja uma característica fundamental para o desenvolvimento de fármacos, propriedades físico-químicas como solubilidade, lipofilicidade e toxicidade também são essenciais para determinar se um composto é "drug-like" (Lipinski et al. 1997)(características físico-químicas que um composto deve ter para ser um candidato a se tornar um medicamento consumível). Nesse contexto, entra a importância da solubilidade e a razão deste trabalho.

Prever solubilidade a partir da estrutura molecular tornou-se um problema relevante em química medicinal e aprendizado de máquina.

Trabalhos como Tetko et al. (2001) e Huuskonen (2000) demonstraram que modelos de regressão linear e redes neurais podem capturar relações úteis entre descritores químicos e valores de solubilidade. Neste trabalho foi utilizado o conjunto de dados apresentado em Applied Predictive Modeling (Kuhn & Johnson, 2013), composto por 1.267 compostos e um conjunto de descritores organizados em três grupos, são eles:

- **208 descritores binários** do tipo "fingerprint", indicando a presença ou ausência de subestruturas químicas específicas.
- **16 descritores de contagem**, relacionados, por exemplo, ao número de ligações ou ao número de átomos halogênios.
- **4 descritores contínuos**, incluindo massa molecular, área de superfície e outras medidas físico-químicas.

Apesar da média das correlações entre os descritores ser baixa, o conjunto de dados possui vários pares de variáveis que são altamente correlacionados, muitas vezes porque alguns descritores medem quase a mesma coisa, como duas versões da área superficial que são idênticas em 87% das moléculas.

O conjunto de solubilidade já vem dividido em treino (951 compostos) e teste (316 compostos). Todo o processo de modelagem, como ajuste dos modelos e validação cruzada, é feito apenas com os dados de treino, enquanto o conjunto de teste fica reservado para a avaliação final. Antes de construir qualquer modelo, investigamos a distribuição dos descritores, suas correlações e como cada um se relaciona com a solubilidade. Como mostrado por Kuhn e Johnson (2013), algumas variáveis têm relação quase linear com o resultado (como peso molecular), mas outras apresentam padrões claramente não lineares, indicando que modelos mais flexíveis podem capturar melhor esses comportamentos.

Com isso, nosso objetivo é comparar vários métodos de regressão como OLS, modelos penalizados (Ridge e Lasso), PCR/PLS e redes neurais e com isso entender qual deles prevê melhor a solubilidade. Usamos RMSE e R^2 como métricas principais, avaliamos a estabilidade via validação cruzada e, no final, discutimos os resultados.

II. Metodologia

Os métodos que norteiam esse trabalho podem ser divididos em cinco partes. A primeira é a análise exploratória dos dados (EDA) bem como a aplicação do pré-processamento. A segunda é a aplicação de métricas de avaliação como RMSE e R^2 para avaliar o modelo da Regressão de Mínimos Quadrados Ordinários (OLS - Ordinary Least Squares) usando os dados de teste e a validação cruzada (CV - Cross Validation). A terceira é aplicar os modelos penalizados como Ridge e Lasso. E por fim, a última é usar os dados treinados para construir Modelos PLS (Partial Least Squares), PCR (Principal Component Regression) e Redes Neurais.

A. Pré-processamento e Análise Exploratória

A eficácia de modelos de regressão linear depende diretamente do atendimento a premissas estatísticas fundamentais, como a linearidade entre preditores e resposta, a normalidade dos resíduos e a ausência de multicolinearidade. A violação dessas condições pode levar a estimativas enviesadas e instabilidade nos coeficientes.

O passo inicial na preparação dos dados consistiu na categorização dos preditores em dois grupos distintos, baseando-se na natureza dos descritores químicos originais:

- **Variáveis Binárias (Fingerprints):** Preditores categóricos que sinalizam a presença ou ausência de subestruturas moleculares específicas.
- **Variáveis Contínuas:** Grupo que engloba tanto os descritores de contagem quanto as medidas contínuas puras, representando propriedades físico-químicas quantitativas.

O foco das transformações matemáticas recaiu sobre as **variáveis contínuas**. Procedeu-se à análise de assimetria (*skewness*), que quantifica o grau de distorção dos dados em relação a uma distribuição normal simétrica. Valores absolutos elevados indicam caudas longas, prejudiciais à regressão OLS.

Para corrigir distorções nas distribuições, optou-se pela aplicação de uma transformação de potência do tipo **Yeo-Johnson**. Este método foi selecionado em detrimento do Box-Cox por sua capacidade de processar valores nulos e negativos, buscando um parâmetro λ ótimo que minimize a assimetria resultante. Simultaneamente, os dados contínuos foram submetidos à padronização (*Z-score*), resultando em variáveis com média zero e desvio padrão unitário.

Adicionalmente, conduziu-se uma investigação visual da linearidade por meio de gráficos de dispersão (*scatter plots*) com curvas de suavização, confrontando os preditores transformados com a variável resposta. Além disso, a estrutura de dependência linear entre as variáveis foi quantificada pela matriz de correlação de Pearson, visualizada via mapa de calor (*heatmap*), estabelecendo o critério numérico para a filtragem de variáveis altamente correlacionadas.

Por fim, para combater a multicolinearidade, implementou-se um algoritmo de filtragem baseado em um limiar de corte (*threshold*). O método percorreu iterativamente a matriz de correlação das variáveis transformadas e, ao identificar pares de preditores com correlação absoluta superior a **0.9**, removeu uma das variáveis do par, preservando apenas os preditores que contribuem com informação única.

B. Validação

1) Modelo OLS (Mínimos Quadrados Ordinários): O OLS (Ordinary Least Squares) é um modelo base que no nosso trabalho foi implementado do zero usando sua fórmula matricial. O objetivo da regressão linear OLS é achar o plano que minimiza a soma dos erros quadráticos (SSE) dada por:

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Sabendo que a forma da regressão linear é parecida com:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + ...$$

Onde:

- y_i = É o valor real que observamos nos dados para a i-ésima observação.
- $\hat{y}_i$ = é o valor previsto pelo modelo para um dado x_i. Não é o valor y_i que observamos, mas sim a estimativa na linha de regressão.
- x = é um dado de entrada observado e fixo.

Os coeficientes β (incluindo o intercepto) são calculados diretamente pela solução de forma fechada:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Onde:

- X = é a matriz das variáveis independentes observadas.
- y = é o vetor de observações da variável dependente.
- $\hat{\beta}$ = é o vetor dos coeficientes estimados.

Esses betas representam o plano otimizado que minimiza os erros quadráticos, ou seja, são os coeficientes que minimizam o erro para determinada feature. Foram implementadas duas versões, uma o OLS foi implementado manualmente e na outra utilizou-se a função pronta do sklearn.

2) Validação Cruzada (CV) e métricas: A validação cruzada k-fold (com k=5) foi implementada em python e funciona da seguinte forma:

Algorithm 1 Validação Cruzada k-fold

Input: Matriz de preditores $X \in \mathbb{R}^{n \times d}$, vetor de respostas $y \in \mathbb{R}^n$, número de *folds* k, função de modelo $model\_fn$.

Output: Médias das métricas RMSE e R^2 ao longo dos k *folds*.

Dividir X e y em k subconjuntos aproximadamente iguais

for $i = 1$ **to** k **do**

  Definir X_{val}, y_{val} como o i-ésimo *fold* Definir X_{train}, y_{train} como os demais $k-1$ *folds*

  Treinar o modelo e obter previsões: $\hat{y}_{val} \leftarrow model\_fn(X_{train}, y_{train}, X_{val})$

  Calcular as métricas do *fold*:

  - $RMSE_i = \sqrt{\frac{1}{m}\sum(y_{val} - \hat{y}_{val})^2}$
  - $R_i^2 = 1 - \frac{\sum(y_{val} - \hat{y}_{val})^2}{\sum(y_{val} - \bar{y}_{val})^2}$

Retornar as médias:

$$RMSE = \frac{1}{k}\sum_{i=1}^{k} RMSE_i, \quad R^2 = \frac{1}{k}\sum_{i=1}^{k} R_i^2$$

TABLE I: Definição de Símbolos e Componentes da Validação Cruzada k-fold

Símbolo	Descrição
$X \in \mathbb{R}^{n \times d}$	**Matriz de Preditoras (Features):** Dados de entrada com n amostras e d características.
$y \in \mathbb{R}^n$	**Vetor de Respostas (Target):** Valores reais que o modelo tenta prever.
k	**Número de Folds:** Número de subconjuntos aproximadamente iguais em que os dados são divididos.
model_fn$(\cdot)$	**Função do Modelo (Callback):** Encapsula o treinamento e a previsão do modelo específico (ex: OLS, Rede Neural).
X_{val}, y_{val}	**Fold de Validação:** O i-ésimo subconjunto usado para testar a performance do modelo.
X_{train}, y_{train}	**Dados de Treinamento:** A união dos $k-1$ folds remanescentes, usados para treinar o modelo na iteração i.
$\hat{y}_{val}$	**Previsões do Modelo:** O vetor de valores previstos pelo modelo, usando X_{val} como entrada ($\hat{y}_{val} \leftarrow$ model_fn$(\cdot)$).
RMSE	**Raiz do Erro Quadrático Médio:** Métrica de erro que mede a magnitude média dos resíduos. Valores menores são melhores.
R^2	**Coeficiente de Determinação:** Métrica de ajuste que indica a proporção da variância em y explicada pelo modelo. Valores mais próximos de 1 são melhores.

C. Modelos Penalizados

Modelos penalizados funcionam de forma semelhante ao OLS tradicional, porém adicionando um termo de penalização λ que reduz a magnitude dos coeficientes β_i, buscando diminuir a sensibilidade do modelo a ruídos. Os modelos utilizados são:

Ridge regression: $\arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$ = Ridge

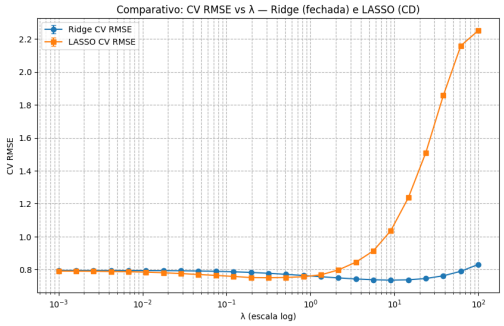
LASSO regression: $\arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$ = LASSO

Os preditores pré-processados foram utilizados para treinar esses modelos. Para cada abordagem, avaliou-se o efeito de λ sobre o desempenho preditivo, utilizando *validação cruzada* k-fold ($k = 5$) apenas no conjunto de treino, evitando *data leakage*.

O procedimento consistiu em:

1) Treinar o modelo para cada λ de um grid previamente definido.
2) Calcular o RMSE e R^2 médio em cada fold, permitindo visualizar o perfil de desempenho e identificar regiões estáveis e instáveis.
3) Selecionar o λ que minimiza o RMSE médio, equilibrando viés e variância.
4) Treinar o modelo final com o λ ótimo no conjunto de treino completo.
5) Avaliar o modelo no conjunto de teste, reportando RMSE e R^2 para verificar generalização.

No Ridge, o RMSE médio de CV tende a diminuir gradualmente com o aumento de λ, indicando uma regularização contínua dos coeficientes sem eliminá-los abruptamente. Já no LASSO, o RMSE apresenta maior sensibilidade e regiões de instabilidade para valores altos de λ, refletindo o efeito de *sparsity* característico da penalização L1. Ambas as contestações estão expostas no gráfico abaixo:



O perfil de RMSE e R^2 obtido por CV fornece uma orientação prática para a escolha de λ, destacando a robustez das implementações manuais e seu comportamento em comparação às funções prontas do `scikit-learn`.

D. PLS/PCR

Dentre as opções de métodos que utilizam a teoria dos componentes principais para reduzir a multicolinearidade dos dados, foi escolhido o PLS. O algoritmo PLS é uma alternativa supervisionada ao PCR. Enquanto o PCR busca direções de variância máxima apenas em $\mathbf{X}$ (não supervisionado), o PLS busca direções que expliquem tanto a variância dos preditores $\mathbf{X}$ quanto a correlação com a resposta $\mathbf{y}$.

Algorithm 2 PLS - Algoritmo NIPALS

Input: Matriz de preditores $X \in \mathbb{R}^{n \times d}$, vetor de respostas $y \in \mathbb{R}^n$, número de componentes M.
Output: $\hat{y}$ e parâmetros do modelo (Φ, Θ, P).

Padronizar X e centralizar y
Inicializar matriz atual $X^{(0)} \leftarrow X$
Inicializar predição acumulada $\hat{y} \leftarrow \bar{y}$
for $m = 1$ **to** M **do**
  Calcular pesos de correlação: $\phi_m \leftarrow (X^{(m-1)})^T y$
  Normalizar pesos: $\phi_m \leftarrow \phi_m / \|\phi_m\|$
  Calcular vetor latente (*score*): $z_m \leftarrow X^{(m-1)} \phi_m$
  Calcular coeficiente $\theta_m \leftarrow \frac{\langle z_m, y \rangle}{\langle z_m, z_m \rangle}$
  Atualizar predição: $\hat{y} \leftarrow \hat{y} + \theta_m z_m$
  Calcular *loadings*: $p_m \leftarrow \frac{(X^{(m-1)})^T z_m}{\langle z_m, z_m \rangle}$
  Atualizar matriz X: $X^{(m)} \leftarrow X^{(m-1)} - z_m p_m^T$
Desfazer a centralização de $\hat{y}$ (somar média de y)
return $\hat{y}$

TABLE II: Definição de Símbolos do Algoritmo PLS (NIPALS)

Símbolo	Descrição
X, y	Matriz de preditores (padronizada) e vetor de respostas (centralizado).
M	Número de componentes latentes (total de iterações).
Operador $\langle \cdot, \cdot \rangle$	Representa o **Produto Interno** (escalar). Em dados centralizados, reflete a covariância.
ϕ_m	**Pesos de Correlação (*Weights*):** Vetor que define a direção de máxima covariância entre X e y.
z_m	**Vetor Latente (*Score*):** O novo componente construído projetando X na direção de ϕ.
θ_m	**Coeficiente de Regressão Interna:** Escalar que define quanto o componente latente z_m contribui para explicar y.
p_m	**Vetor de Carga (*Loading*):** Coeficiente de projeção usado para calcular quanto de X foi explicado por z, permitindo a deflação.
$\hat{y}$	**Predição Acumulada:** Vetor contendo a soma das contribuições de todos os componentes calculados até o momento.

O algoritmo acima foi implementado em `Python`.

E. Rede Neural

As Redes Neurais são modelos computacionais inspirados no sistema nervoso biológico, capazes de aprender e modelar relações não-lineares complexas entre variáveis de entrada (preditores) e saída (resposta). O modelo mais comum para regressão é o *Multilayer Perceptron* (MLP).

A unidade fundamental de processamento é o neurônio artificial. Matematicamente, o funcionamento de um neurônio k pode ser descrito em duas etapas:

1) **Combinação Linear (v_k):** O neurônio calcula uma soma ponderada dos sinais de entrada x_j, ajustada por um viés (*bias*) b_k.

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k \tag{1}$$

Onde w_{kj} é o peso sináptico que conecta a entrada j ao neurônio k.

2) **Ativação (y_k):** O campo local induzido v_k é passado por uma função de ativação não-linear $\varphi(\cdot)$ para gerar o sinal de saída.

$$y_k = \varphi(v_k) \tag{2}$$

Exemplos comuns de $\varphi(\cdot)$ incluem a sigmoide ($\frac{1}{1+e^{-v}}$) e a ReLU ($\max(0, v)$).

Algoritmo de Aprendizagem - *Backpropagation*: O treinamento de uma rede neural é um problema de otimização que visa minimizar uma função de custo E (geralmente o Erro Quadrático Médio - MSE) em relação aos pesos w. O algoritmo *Backpropagation* utiliza a regra da cadeia para calcular o gradiente do erro.

Forward Pass: Para cada exemplo de treino, o sinal flui da camada de entrada, passando pelas camadas ocultas, até gerar a predição na camada de saída.

Backward Pass: Calcula-se o erro na saída e propaga-se esse erro de volta para atualizar os pesos. O ajuste é feito através da **Regra Delta**:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = \eta \delta_j y_i \tag{3}$$

Onde:

- η (*learning rate*): Taxa de aprendizagem.
- δ_j: Gradiente local do neurônio j.

Cálculo do Gradiente Local (δ_j):

- **Para neurônios de saída:** Depende diretamente do erro de predição ($e_j = d_j - y_j$).

$$\delta_j = e_j \varphi'(v_j) \tag{4}$$

- **Para neurônios ocultos:** O erro é retropropagado das camadas seguintes (k).

$$\delta_j = \varphi'(v_j) \sum_k \delta_k w_{kj} \tag{5}$$

Inicialmente, os dados foram submetidos a um pré-processamento de padronização (Z-score), onde as estatísticas (média e desvio padrão) foram calculadas exclusivamente no conjunto de treino e aplicadas ao conjunto de teste para evitar vazamento de dados.

A arquitetura da rede neural consiste em um Perceptron Multicamadas (MLP) com uma única camada oculta contendo 5 neurônios. Este número foi escolhido para equilibrar a capacidade de capturar não-linearidades sem introduzir complexidade excessiva. O treinamento foi realizado via Gradiente Descendente por 5000 épocas com uma taxa de aprendizado (learning rate) de 0.01, garantindo a convergência da função de custo. Para mitigar o risco de overfitting, aplicou-se uma regularização L2 (weight decay) de 0.001, penalizando pesos de grande magnitude.

III. Resultados e Discussão

A. Resultados - EDA e Pré-Processamento

A análise inicial das distribuições revelou que diversos descritores apresentavam distribuições severamente distorcidas. Variáveis como **NumSulfer** (contagem de átomos de enxofre) e **HydrophilicFactor** exibiram coeficientes de assimetria de 3.8476 e 3.40916, respectivamente, indicando forte assimetria positiva.

Após a aplicação do pipeline de pré-processamento (Yeo-Johnson e padronização), o cálculo da *skewness* foi refeito para validar a eficácia do método. Observou-se uma melhoria significativa na simetria das distribuições: a variável **NumSulfer** teve sua assimetria reduzida para 2.24079, uma correção expressiva dada a natureza discreta da variável, enquanto o **HydrophilicFactor** atingiu um valor de 0.250059, aproximando-se consideravelmente da normalidade.

A inspeção visual através de gráficos de dispersão revelou heterogeneidade nas relações. Preditores como **MolWeight** demonstraram forte linearidade com a solubilidade. Em contrapartida, variáveis associadas a contagem de átomos, como **NumHydrogen**, exibiram comportamentos não-lineares, sugerindo que modelos lineares rígidos (como OLS) podem subestimar a complexidade dessas relações.

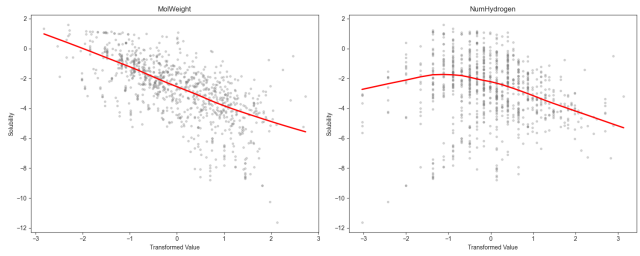


Fig. 1: Análise visual de linearidade: comparação entre a relação aproximadamente linear de (*MolWeight*) e o comportamento não-linear de (*NumHydrogen*) com a Solubilidade.

A Fig. 2 apresenta a matriz de correlação de Pearson para as variáveis contínuas transformadas. O mapa de calor evidencia agrupamentos de variáveis com correlação cruzada elevada, confirmando a existência de uma multicolinearidade significativa que poderia inflar a variância das estimativas.

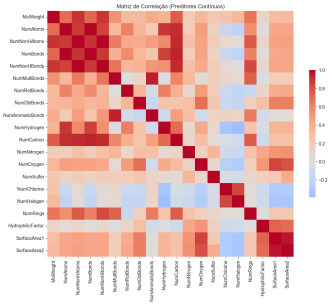


Fig. 2: Matriz de correlação de Pearson.

Após a filtragem (correlação > 0.9), o conjunto de preditores foi reduzido de 228 para 192 variáveis, ou seja, foram removidas 36 variáveis, diminuindo a complexidade computacional e o risco de multicolinearidade severa.

Ao final desta etapa, os dados apresentavam-se transformados, padronizados e filtrados, fornecendo uma base robusta para a aplicação dos modelos de regressão.

B. Resultados - Validação

Nesta etapa avaliamos o desempenho do modelo de regressão linear OLS utilizando os descritores transformados do conjunto de treinamento, comparando a implementação manual, baseada na solução matricial fechada, e a implementação disponível na biblioteca `scikit-learn`.

Ambas as formas de implementação foram ajustadas usando o conjunto de treinamento, e avaliados no conjunto de teste por meio das métricas RMSE e R^2. Obtemos então esses resultados:

TABLE III: Desempenho dos modelos OLS no conjunto de teste.

Modelo	**RMSE (Teste)**	**R^2 (Teste)**
OLS Manual	0.7405	0.8727
OLS *scikit-learn*	0.7405	0.8727

Percebemos que ambos o OLS manual e o usado pelo sklearn deram exatamente os mesmos valores. Isso se dá pelo fato de termos implementados a fórmula certa além de adicionar a coluna de 1s e a padronização dos dados, já que o OLS é sensível a escalas. Com isso a gente percebe que o dataset não apresentou problemas de multicolinearidade severa, apresentou singularidade em $X^T X$ e variáveis quase constantes. Se algo tivesse acontecido os valores poderia ser diferentes, ou seja, reproduzimos perfeitamente o método do sklearn.

Dos valores tiramos que em média nosso modelo erra 0.74 unidades ao prever valores de solubilidade e o modelo explica 87% da variância, o que é um valor alto mostrando ser um bom modelo no teste.

Agora, analisemos os resultados da validação cruzada com $k = 5$ para os dois modelos OLS.

TABLE IV: Resultados da validação cruzada (5-fold) para os modelos OLS.

Modelo	**RMSE (CV)**	**R^2 (CV)**
OLS Manual	0.7919	0.2950
OLS *scikit-learn*	0.7919	0.2950

Percebe-se logo que o RMSE da cross-validation (0.79) ficou um pouco maior que o RMSE do teste (0.74), enquanto o R² despencou de 0.87 no teste pra 0.29 na CV. O RMSE não variou tanto, então não parece ter overfitting, mas isso também mostra que a divisão original do dataset, que já veio pré-separado, provavelmente foi melhor do que os splits aleatórios da CV. Pode ser que algum fold tenha ficado fora da distribuição normal dos dados. Já o R² caiu demais, mostrando que o modelo fica instável dependendo de como os dados são

divididos. Isso pode acontecer porque temos muitos preditores pra pouca amostra, então alguns folds acabam com padrões bem diferentes.

Pra resolver isso e deixar o modelo mais estável, dá pra usar técnicas como Ridge, Lasso, PCR, PLS e afins.

No geral, o OLS funcionou bem: a versão manual bate idêntico com a do scikit-learn e o modelo teve bons RMSE e R^2 no teste, capturando bem a relação entre as variáveis e a solubilidade. Mas a CV deixou claro que existe instabilidade, talvez por colinearidade ou efeitos não lineares. Então o OLS serve mais como um ponto de partida sólido pra comparar com métodos mais avançados no resto do trabalho.

C. Resultados - Modelos Penalizados

Nesta análise, implementamos modelos Ridge e LASSO de forma manual, avaliando-os com validação cruzada e comparando-os com implementações prontas. Observamos que:

- O Ridge apresentou comportamento estável, com RMSE decrescendo suavemente à medida que λ aumentava, controlando a magnitude dos coeficientes sem eliminá-los abruptamente.
- O LASSO mostrou comportamento mais sensível a λ, podendo zerar coeficientes e causar instabilidades numéricas se o parâmetro não for bem calibrado.
- A escolha do λ ótimo via validação cruzada permitiu equilibrar o *trade-off* viés-variância, garantindo bons resultados no test set, com RMSE e R^2 consistentes com o observado durante a CV.

Em síntese, a experiência prática evidenciou como a regularização influencia a generalização do modelo, destacando os pontos fortes e limitações de cada abordagem e fornecendo compreensão sobre a magnitude dos coeficientes e sparsity introduzida pelo LASSO.

D. Resultados - PCR/PLS/Rede Neural

Para avaliar a capacidade preditiva dos modelos desenvolvidos, comparou-se o desempenho do método PLS com o método não-linear de Redes Neurais no conjunto de teste independente.

1) Desempenho do Modelo Linear (PLS): O modelo PLS foi otimizado via validação cruzada (*k-fold* $= 10$), resultando na escolha de 40 componentes latentes para capturar a variância explicativa dos dados. Ao aplicar este modelo ao conjunto de teste, obteve-se um erro quadrático médio (RMSE) de **0.7425** e um coeficiente de determinação (R^2) de **0.8720**.

A análise dos gráficos de validação cruzada demonstra que a estrutura dos dados exige um número elevado de variáveis latentes para ser modelada linearmente. Observa-se uma melhoria significativa na performance preditiva (queda do RMSE e aumento do R^2) até aproximadamente 15 componentes. No entanto, o erro continua a decair marginalmente, atingindo seu mínimo global em 40 componentes. Portanto, selecionou-se $M = 40$ para maximizar a acurácia, aceitando a maior complexidade computacional em troca da minimização do erro residual.

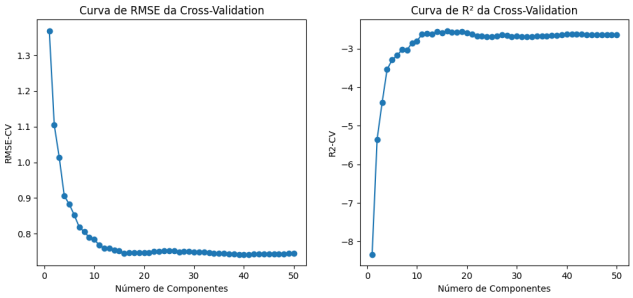


Fig. 3: Cross-validation profile

2) Desempenho do Modelo Não-Linear (Rede Neural): A rede neural foi configurada com uma arquitetura de 5 neurônios na camada oculta, utilizando ativação sigmoide e regularização *weight decay* ($\lambda = 0.001$) para controlar a complexidade e evitar *overfitting*. Após 5000 épocas de treinamento, o modelo apresentou no conjunto de teste um RMSE de **0.7178** e um R^2 de **0.8804**.

Ao analisar os resultados consolidados na Tabela V, observa-se a superioridade do modelo não-linear.

TABLE V: Comparação de desempenho dos modelos no conjunto de teste.

Modelo	**RMSE (Teste)**	**R^2 (Teste)**
PLS (40 componentes)	**0,7425**	**0,8720**
Rede Neural (5 neurônios)	**0,7178**	**0,8804**

A comparação direta demonstra que a Rede Neural superou o PLS em ambas as métricas de avaliação. O modelo neural reduziu o erro de predição (RMSE) e aumentou a variância explicada (R^2).

Este resultado sugere que a relação subjacente entre os descritores moleculares e a solubilidade não é estritamente linear. Enquanto o PLS tenta projetar os dados em um subespaço linear latente (exigindo 40 componentes para atingir seu melhor desempenho), a rede neural conseguiu capturar interações complexas e não-lineares entre as variáveis utilizando uma representação interna mais compacta (apenas 5 neurônios ocultos).

A capacidade da rede neural de modelar essas não-linearidades permitiu uma generalização superior, confirmando a vantagem de métodos de aprendizado de máquina não-lineares para este tipo de problema.

Referências

[1] M. Kuhn and K. Johnson, Applied Predictive Modeling. New York, NY, USA: Springer, 2013.

[2] A. Leach and V. Gillet, *An Introduction to Chemoinformatics*. Dordrecht, Netherlands: Springer, 2007.

[3] I. Tetko, V. Tanchuk, T. Kasheva, and A. Villa, "Estimation of Aqueous Solubility of Chemical Compounds Using E–State Indices," *Journal of Chemical Information and Computer Sciences*, 2001.

[4] C. Lipinski, F. Lombardo, B. Dominy, and P. Feeney, "Experimental and Computational Approaches To Estimate Solubility and Permeability In Drug Discovery and Development Settings," *Advanced Drug Delivery Reviews*, 1997.