Trabalho 2 - INF-0615

Kaleb Roncatti de Souza e Nelson Gomes Brasil Júnior

11 de setembro de 2022

1 Introdução

A inteligência artificial tem se tornado de mais em mais comum no dia-a-dia da nossa sociedade. Dentre suas múltiplas aplicações, uma das mais nobres/importantes está relacionada à área da saúde. No trabalho atual, analisaremos um dataset que possui informações/features de cadeias de proteínas e o objetivo principal é predizer se ela estimula ou não a produção de anticorpos no organismo humano. Ao longo deste trabalho, usaremos modelos de regressão logística para classificação binária, um modelo de aprendizado de máquina **supervisionado**, analisando-se combinações entre features.

2 Tarefas

2.1 Inspeção dos dados

Assim como trabalho anterior, começamos carregando os *datasets* de treino, validação, e ambos os conjuntos de teste (comum e SARS). Observe que, o objetivo de carregarmos os conjuntos de teste logo no início do código se mostra apenas na realização de transformações (tais como normalização) similares que serão aplicadas ao conjunto de treino e validação, para que os modelos fiquem consistentes no momento de realizarmos previsões, porém, os conjuntos de teste seguem intactos de qualquer técnica de treinamento.

Ao inspecionarmos o conjunto, **não observamos** nenhum modelo que não estivesse anotado, ou seja, todos os dados possuem um `target`. Porém, caso tivéssemos observado a falta de anotações, poderíamos fazer como explicado no trabalho anterior: preencher os dados com a média ou mediana dos dados de uma dada feature feature (ou um agrupamento de dados da feature). Dependendo da quantidade e da relevância dos dados faltantes, poderíamos removê-los sem nenhum causar nenhum problema no modelo.

Observamos a seguinte quantidade de amostras por conjunto: `9204` amostras de treino, `2303` amostras de validação, `2878` amostras de teste e `520` amostras de teste do tipo SARS.

2.2 Frequência das classes de um `target` e balanceamento

No que diz respeito ao balanceamento, caso não seja propriamente tratado, pode gerar um `bias` grande no modelo, dado que a técnica acaba entendendo que uma classe/target pode ser mais importante em detrimento à outra. Idealmente, o fato de existirem números distintos de amostras de treinamento, não necessariamente implicará que o modelo de fato precisa ser

rebalanceado, mas já se demonstra como um pertinente indício de tal ocorrência. O desbalanceamento fica mais evidente quando visualizamos a matriz de confusão após treinamento/predição do modelo já para o conjunto de treino. Sendo assim, percebemos que, já base de treino, temos `6709` amostras com `target = 0` (72.8% dos dados) e `2495` amostras com `target = 1` (27.1% dos dados). Desta forma, optamos por usar a técnica de balanceamento de pesos (`weighting`), a qual impõe um peso na função de custo para a classe cujo `target` está desbalanceado (com menos amostras).

2.3 Normalização

Para tal trabalho, não precisamos lidar com features do tipo categóricas dado que todas eram contínuas, então manipulamos as features numéricas através de uma normalização, de modo a não termos valores muito discrepantes entre si os quais possivelmente enviesariam nossos modelos, e desta forma prontos para o treinamento. Assim como no trabalho anterior, escolhemos a técnica da `Z-Norma`, a qual calcula a média (μ) e o desvio padrão (σ) para cada feature do conjunto de treinamento aplicando a transformação aos dados de treino, validação e teste. Note que, ao aplicar esta transformação aos dados de treino, todas as features terão média 0 e variância 1 e, se os dados seguirem uma distribuição normal padrão, nada ocorreria. Outrossim, para todos os datasets, treino, validação e testes, mantivemos os valores de μ e σ do conjunto de treino.

2.4 Regressão Logística/Sigmoide - `Baseline Model`

Como modelo considerado `baseline`, utilizamos todas as features fornecidas para o treinamento num modelo de regressão logística. A seguir, podemos observar o caráter da função sigmoide (Equação 1) utilizada em modelos de previsão categórica e supervisionada.

$$\sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}} \tag{1}$$

Para tal modelo compondo todas as features e utilizando-se $\lambda = 10^{-6}$ e os pesos na função de custo explicitados anteriormente, chegamos na seguinte matriz de confusão relativa (valores `reference` nas linhas e `predictions` nas colunas).

Através da matriz, conseguimos extrair os valores de `TPR = 0.6457` e `TNR = 0.6016`, assim como a `balanced accuracy = 0.6236` para o conjunto de **treino**. `TPR = 0.648` e `TNR = 0.5852`, assim como a `balanced accuracy = 0.6166` para o conjunto de **validação**. `TPR = 0.6415` e `TNR = 0.5961`, assim como a `balanced accuracy = 0.6188` para o conjunto de **testes**.

2.5 Regressão Logística/Sigmoide - `Playing around`

Nesta seção, usamos algumas combinações de features e variamos o grau dos polinômios para gerar novos modelos e assim poder analisar os resultados e identificar regiões de *underfitting*, *região ótima* e *overfitting*.

Para estas análises e pensando em extrair o melhor resultados dos dados, utilizamos a mesma técnica do trabalho anterior, inspecionando a matriz de correlação entre as variáveis, dado que

possuímos apenas features que se mostram contínuas. O `plot` da correlação é apresentado na Figura 1 a seguir.

Conseguimos observar que as features `start_position` e `end_position` se mostram quase que 100% correlacionadas, desta maneira, deixamos de fora do nosso modelo a característica `end_position`.

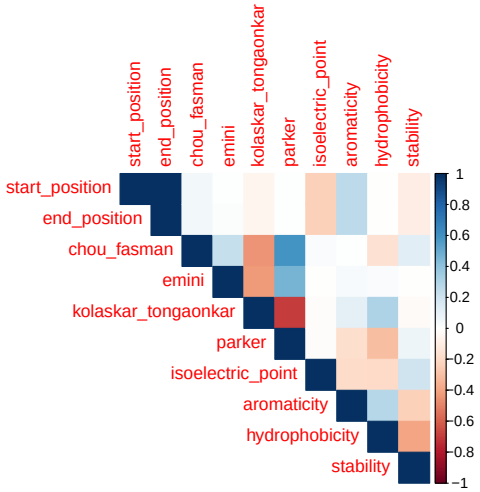

Figura 1: Correlação entre as features numéricas

Uma análise parecida foi realizada com as features entre `chou_fasman` e `parker`, o que nos levou a deixar a feature `parker` de fora do modelo também.

2.5.1 Polynomial models

Nosso primeiro passo foi utilizar o modelos polinomial com as features escolhidas, variando o grau do polinômio de 0 a 15. O resultado está apresentado na Figura 2. Observamos a partir do gráfico que quando aumentamos o grau do polinômio a acurácia balanceada nos conjuntos de treino e validação apresentam um comportamento crescente até grau 6 e mantém uma certa estabilidade para graus maiores. Então podemos concluir que, para polinômios com grau menor do que 5 temos uma região de underfitting e, acima deste valor temos a região ótima, não havendo overfitting neste caso. Porém, se observarmos mais atentamente para um grau de complexidade elevado, a partir do grau 13, conseguimos ver que a acurácia balanceada de validação volta a aumentar. Uma análise mais profunda seria ideal (avaliando-se para valores ainda mais elevados) para o caso corrente, no entanto, algo para se levar em consideração se mostra como o poder computacional para tais modelos, o qual também aumenta conforme aumentamos o grau polinomial. O melhor modelo polinomal (grau 15) demonstra `TPR = 0.6914`, `TNR = 0.6624` e `balanced accuracy = 0.6769` para o conjunto de **testes**.

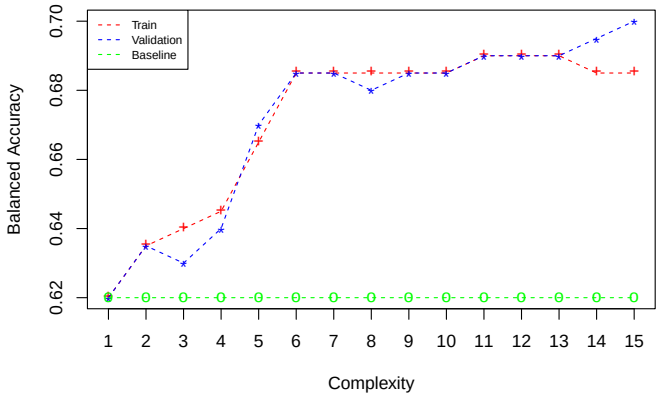

Figura 2: Evolução da acurácia balanceada de acordo com o grau dos polinômios

2.5.2 Combining features

Para a combinação de features, fizemos um processo similar ao que fizemos no caso polinomial, ou seja, realizamos uma combinação dos modelos com a combinação entre features no `R` variando-se de 0 até 15. O resultado está na Figura 3. Da mesma forma que o caso polinomial, aqui vimos uma estabilidade nos valores de acurácia balanceada para graus altos (a partir da combinação de grau 6), levando a um resultado de underfitting para graus abaixo de 4 e a região ótima para valores acima deste. Novamente não observamos regiões de overfitting.

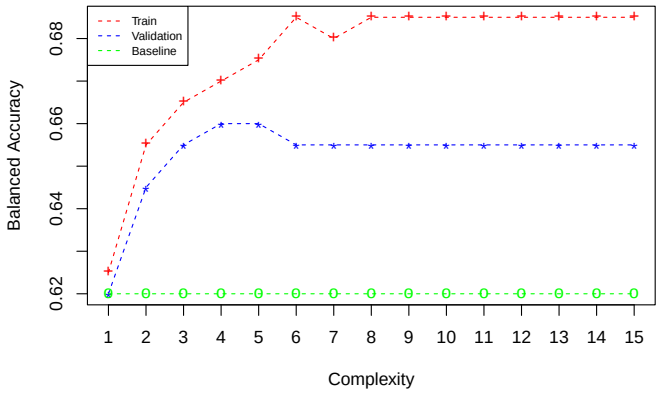

Figura 3: Acurácia balanceada de acordo com a complexidade da combinação entre as variáveis

Para este modelo, obtivemos os seguintes resultados para o conjunto de testes: `TPR = 0.6658` e `TNR = 0.6481`, assim como a `balanced accuracy = 0.6569`.

2.5.3 Regularização

Selecionando o modelo de combinação de features de grau quatro, modelo que exige poder computacional relativamente baixo e apresenta resultados adequados com relação aos outros modelos testados, fizemos uma análise do parâmetro λ da regularização. Para isto, variamos este parâmetro com 10^i, $i = -7 : 1$. O resultado obtido está na Figura 4. Podemos observar que a medida que o parâmetro cresce, a acurácia balanceada tanto no conjunto de validação como de treinamento diminui rapidamente. Assim, podemos definir que a região de underfitting é $\lambda \geq 0.01$ e a região ótima é quando tomamos $\lambda \leq 10^{-3}$, sendo o gráfico bem parecido, ainda que espelhado, com os resultados vistos nas seções anteriores.

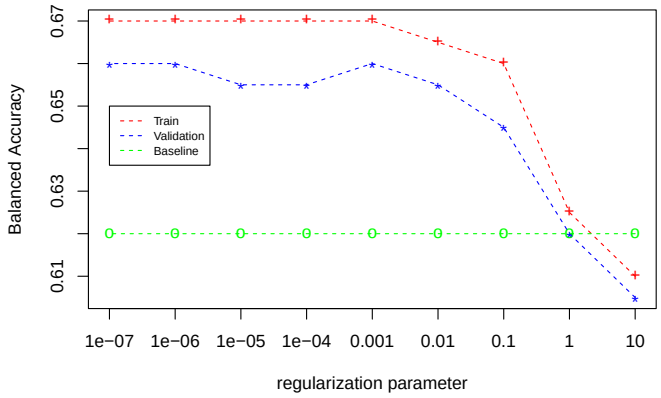

Figura 4: Acurácia balanceada de acordo com o parâmetro de regularização λ

Escolhemos então o modelo com $\lambda = 10^{-3}$ e calculamos o seu resultado no conjunto de testes, obtendo `TPR = 0.6671` e `TNR = 0.6443`, assim como a `balanced accuracy = 0.6557`.

2.6 Conclusão

Combinando os resultados das duas seções anteriores (modelos **polinomiais** e **combining features**, observamos que o melhor modelo foi o polinomial de grau 15, ou seja este foi o modelo que apresentou maior acurácia balanceada no conjunto de validação.

Comparando os resultados no conjunto de *SARS_test_csv*, escolhemos o modelo polinomial de grau 15 como o mais performante, e tivemos `TPR = 0.6643` e `TNR = 0.4132`, assim como a `balanced accuracy = 0.5387`. Notamos que para este conjunto o resultado se mostrou inferior aos conjuntos de teste, validação e treino, se aproximando de um classificador praticamente aleatório ao demonstrar acurácia perto de 0.5. Uma possível razão para tal, seria a não existência ou a existência de um valor quase que irrelevante de dados de treino que remetessem à tal vírus, causando uma performance deturpada quando aplicado o modelo que treinamos. Porém, não podemos descartar a possibilidade de simplesmente o modelo ter performado mal por falta de `tuning` nos parâmetros que utilizamos ou falta de refinamento nos modelos em si.