Relatório 1: Linear vs MLP

João Victor de Souza Albuquerque

[1]Instituto Federal de Educação, Ciência e Tecnologia do Ceará - Campus Maracanaú (IFCE)
Av.Parque Central, 1315 - Distrito Industrial I, Maracanaú - CE, 61939-140

Resumo. *Este relatório documenta o experimento do primeiro trabalho de rna.*

1. Base Teórica

1.1. Regressão Logística

Os problemas de predição não se resumem apenas a prever valores contínuos, outro problema bem comum é o de classificação, que consiste a partir de padrões comuns a uma classe ou diferentes classes identificar uma amostra desconhecida em uma classe conhecida. Nos modelos anteriores as predições nos davam valores contínuos, e para classificação é necessário que as saídas sejam discretas e que sejam valores que representam as classes, no caso da classificação logística trabalharemos com valores entre 0 e 1, então é necessário fazer uma modificação na equação 1 para que ela nos forneça valores nessa faixa, para isso se encontrou a solução de aplicar a função sigmóid na saída da equação 1 que nos gera a equação 2:

$$\hat{y}_i = w^T x_i, \tag{1}$$

$$\hat{y}_i = \sigma(w^T x_i), \tag{2}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}. \tag{3}$$

Com a aplicação dessa nova ideia para a criação de um classificador, é necessário reformular as regras de atualização dos Ws e a da função de perda do modelo. A ideia é que ao minimizar as perdas do modelo, os Ws sejam atualizados ao ponto que o y predito sejam valores muito próximos de 0 ou 1, para isso se utiliza a função de perda de entropia cruzada, equação 4, e a função de atualização dos Ws na equação 5:

$$\mathcal{J}(w) = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log \sigma(w^T x_i) + (1 - y_i) \log(1 - \sigma(w^T x_i)) \right], \tag{4}$$

$$w(t) = w(t-1) + \alpha e_i(t-1) x_i. \tag{5}$$

A ultima mudança necessária é na montagem do modelo é adicionar uma função limiar nas saídas da predição, pois as classes são 1 ou 0 e o calculo de y predito nos retorna um valor próximo desses dois extremos, portanto foi aplicado uma função limiar que valores preditos como menores ou igual a 0.5 retornam 0 e valores maiores que 0.5 retornam 1.

1.2. Normalização

As vezes os dados que queremos que o modelo se adapte estão em escalas muito diferentes o que pode acarretar em dificuldade do modelo em se adaptar a eles, por isso é recomendado deixar os dados numa escala comum, esse processo se chama de normalização, que pode ser colocar os dados em uma escala de [0,1] ou [-1,1] ou mesmo forção uma média e variância. Esse método de preprocessamento nos da maior controle dos valores dos parâmetros e facilita o ajuste dos hiperparâmetros

No trabalho foi utilizado a normalização dos dados na escala de [0,1], essa regularização seguir os seguintes passos:

Primeiro se calcula o valor máximo do vetor Y e matriz X, coluna a coluna:

$$y_{max} = \max(y), \quad [x_{max}]_d = \max([X]_d), \forall d.$$

Depois disso, calcule o menor valor do vetor Y e da matiz X, coluna a coluna:

$$y_{min} = \min(y), \quad [x_{min}]_d = \min([X]_d), \forall d.$$

E por fim se aplica a formula a seguir para finalizar a regularização:

$$y \leftarrow \frac{y - y_{min}}{y_{max} - y_{min}}, \quad [X]_d \leftarrow \frac{[X]_d - [x_{min}]_d}{[x_{max}]_d - [x_{min}]_d}, \forall d.$$

2. Metodologia

2.1. Dados

Importei os dados contidos no breast.csv e os dados da Iris dataset da biblioteca do sklearn, ambos os datasets foram salvos em dataframes do pandas. O dataset da breast não passou por nenhuma modificação nos seus dados, já o dataset da Iris passou por algumas modificações, a primeira modificação foi cortar as colunas de features para que seja utilizado apenas as features petal length, petal width como o X, a outra modificação foi juntar as classes versicolor e virginica em apenas uma classe 0 e transformar a classe setosa na classe 1 para que o meu problema se tornasse um problema binário.

2.2. Adaptações do modelo

Para transformar o modelo de regressão polinomial existente em um classificador logístico foi necessário fazer uma modificação na função fit, trocando o chamamento interno de fit_ols() para fit_sdg(), ja no fit_sdg() foi necessário refaze-la toda para codificar as equações de predição de y e equações de atualização dos Ws, além disso foi necessário refazer a função de loss para que ela fosse a função de entropia cruzada.

Outra modificação feita na função Predict() que atualizando a equação de predição para que ela se encaixasse com a equação já descrita acima da logística e foi adicionado uma função limiar que que aproxima o valor de y predito para 0 ou 1 antes de retornar a classificação.

2.3. Normalização

Para Implenetar a classe de normalização, preciso desenvolver 3 funções, a função fit(), normalize e desnormalize. A função fit salva os minemos e máximos do X e Y internamente na classe. A função de normalize pega os valores de máximos e minemos salvos na classe e utiliza na função de normalização já mostrada e que foi traduzida para código, e no final devolve os X e Y normalizados. A função desnormalize pega os valores mínimo e máximo salvos internamente na classe e as utiliza na inversa da função de normalização.

2.4. Questão 1

Primeiro separei os dados em conjunto de treino e teste, utilizando a função train_test_split() do sklearn com test_size de 0.2 e random_state de 42. Após isso, separei o conjuto de treino em train_x e train_y, e o conjunto de teste em test_x e test_y.

Com os conjuntos já em mãos, eu defini a classe de normalização e chamei sua função fit() para ajustar a escala aos dado. Em seguida chamei a função normalize para o conjunto de treino e teste, e salvei as saídas em variáveis.

Com os dados já normalizado, fiz a instanciação do modelo, o treinei e fiz a predição com os dados de teste, após isso, fiz o calculo da quantidade de verdadeiros positivo, falsos negativo, falsos positivo, e verdadeiro negativo. Em seguida fiz os cálculos das estatísticas do modelo, acurácia, revocação, precisão e f1-score, por ultimo plotei a matriz de confusão.

Em seguida ao primeiro teste, eu o refiz modificado os hyperparametros para ter uma predição não linear.

Por ultimo fiz um teste das estatísticas do modelo utilizando 10 folds, com o auxilio da função kfold() do sklearn, em cada rodada do fold eu fazia os mesmo passo a passo do primeiro teste do modelo com exceção em plotar a matriz de confusão e do split. No fim de todas as rodadas fiz uma media e desvio padrão de cada uma das estatísticas.

2.5. Questão 2

Primeiro fiz a separação dos dados do dataset iris adaptado, separei eles em treino e teste com biblioteca split do sklearn e fiz uma separação estratificada devido ao desequilíbrio dos dados. Depois disso treinei o modelo com os dados de teste, fiz uma predição com os dados de teste e exibi a comparação no terminal e por ultimo plotei um gráfico de cores do modelo.

3. Resultado

3.1. Questão 1

Comparado as estatísticas do teste com o modelo logístico com coluna linear, imagem 1, e do modelo logístico com coluna não linear, se ver que o modelo não linear tem um performasse melhor na revocação e acurácia, e as outras estatísticas continuam igualmente ótimas. Em relação ao teste do kfold, obtive o resultado da imagem 3, que se ver que os valores da media foram menores que o resultado obtido nos teste unitários.

3.2. Questão 2

Como resultado do treino do modelo com os dados da iris, obtivemos um modelo perfeito para esses dados como pode ser visto pelas estatísticas da imagem 4, tendo resultados perfeitos em todos os parâmetros. E fazendo o mapa de cores para o modelo se ver o resultado da imagem 5.

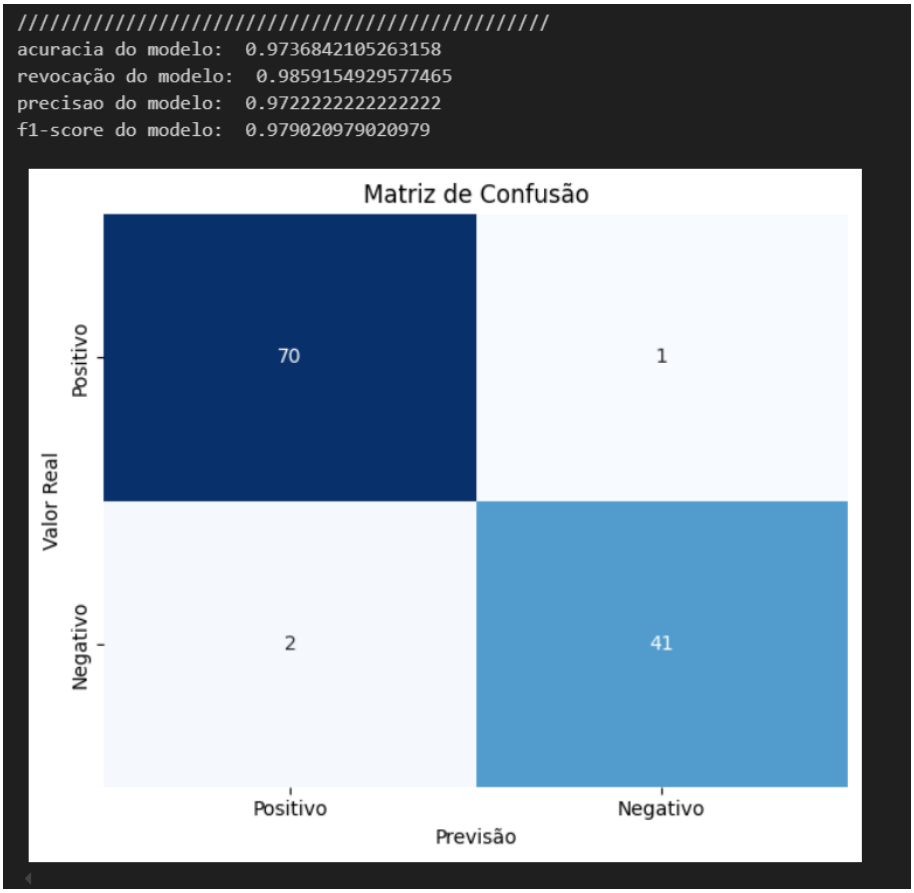

Figure 1. modelos linear

```
////////////////////////////////////////////////
acuracia do modelo:  0.9824561403508771
revocação do modelo:  1.0
precisao do modelo:  0.9726027397260274
f1-score do modelo:  0.9861111111111112
```

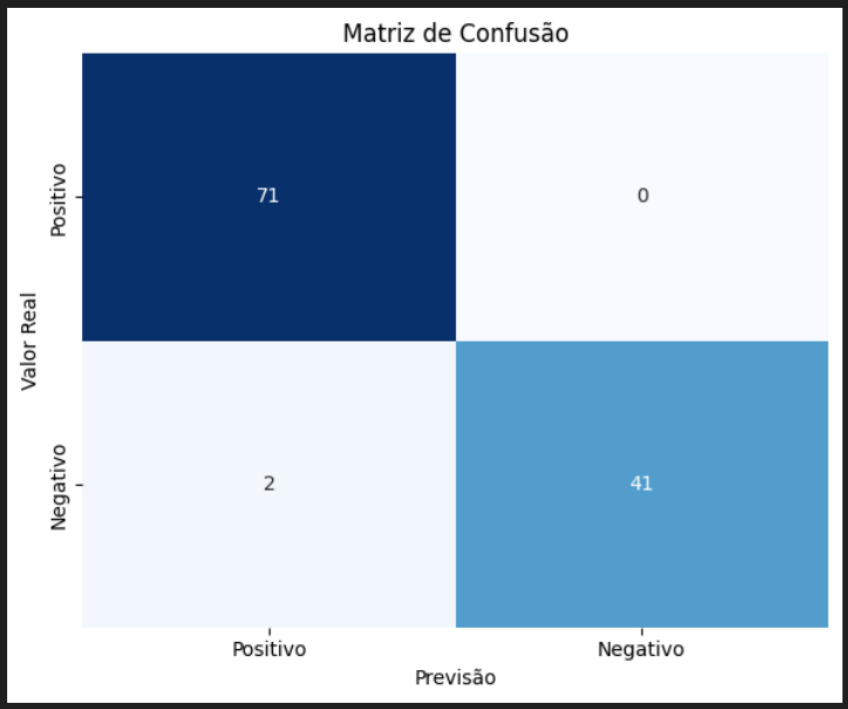

Figure 2. modelos não-linear

```
////////////////////////////////////////////////////
Metricas Acuracia:
Acuracia médio: 0.8839912280701755
Desvio padrão Acuracia: 0.04852837791028952

Metricas Revocação:
Revocação médio: 0.9354225348176961
Desvio padrão Revocação: 0.036821315818628l7

Metricas Precisão:
Precisão médio: 0.8847500541820184
Desvio padrão Precisão: 0.07449830905049408

Metricas F1-score:
F1-score médio: 0.9073301239307314
Desvio padrão F1-score: 0.04306669777286206
////////////////////////////////////////////////////
```

Figure 3. 10 k-folds

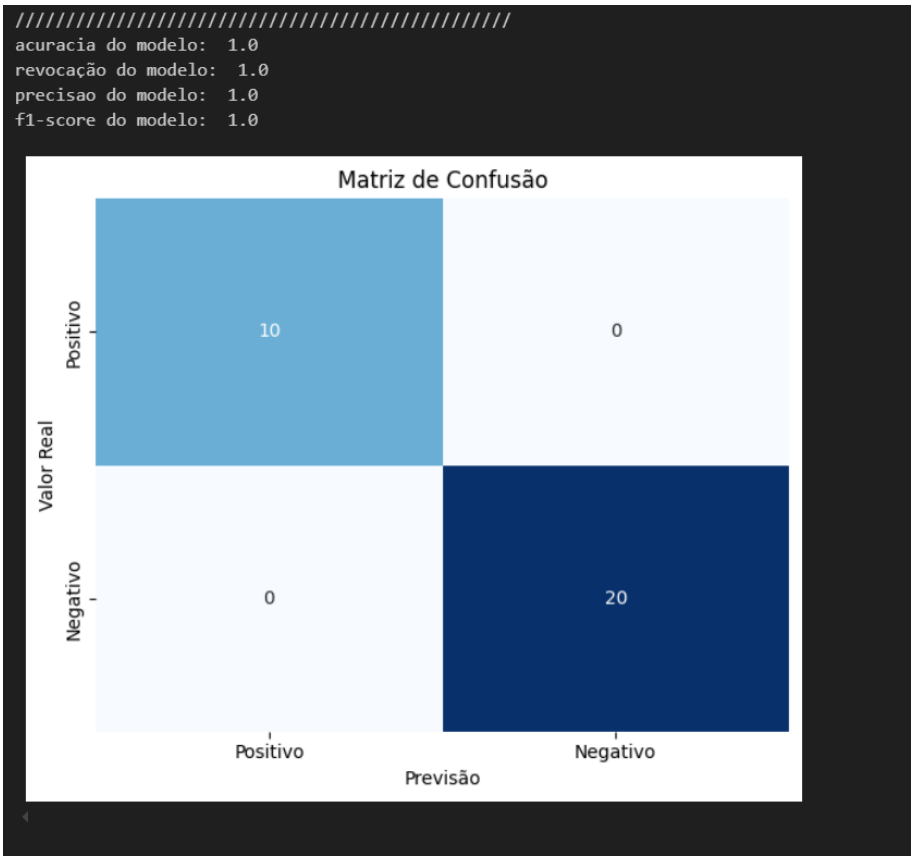

Figure 4. modelo com os dados da iris

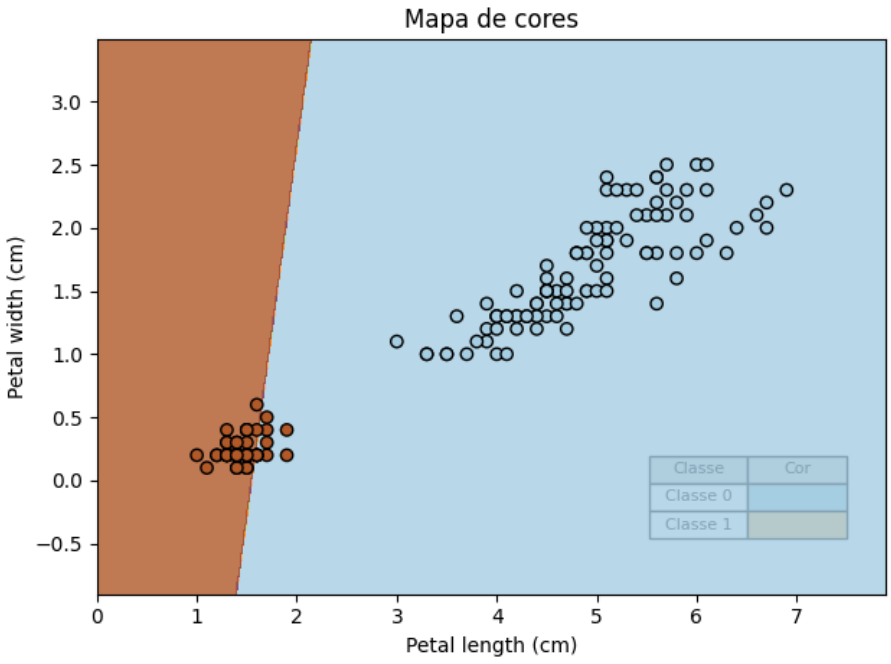

Figure 5. mapa de cores