Universidade Federal de Pernambuco
Centro de Informática

Atividade Individual 2

Avaliação de algoritmos de aprendizagem

Aluno: Karl Vandesman de Matos Sousa
Professor: Leandro Maciel Almeida

1 Introdução

Esta prática trata da aplicação de algoritmos de aprendizagem em um conjunto de dados sobre doença da tireoide (*Thyroid Disease Data Set - UCI Machine Learning Repository*). Na atividade anterior, já se havia trabalhado com esses dados, por meio de análise exploratória e pré-processamento. Dessa forma, partiu-se do código desenvolvido anteriormente para então avaliar o desempenho de três algoritmos de aprendizagem, por meio da acurácia medida.

Antes de entrarmos nos algoritmos de aprendizagem selecionados, é interessante fazer algumas observações acerca da base de dados em que se está lidando. Nesse caso, temos uma base bastante desbalanceada com menos de 10% das observações com resultado positivo (presença da doença). Dessa forma, se o modelo do classificador sempre classificar uma observação como sendo negativa, já teríamos uma acurácia com um valor, a princípio, atraente, maior que 90%. Logo, nesse contexto a acurácia não seria a melhor métrica de desempenho a ser utilizada. Outras opções de métrica poderiam ser o *recall* e a f1 *score*:

$$recall = \frac{Verdadeirospositivos(TP)}{Verdadeirospositivos(TP) + Falsosnegativos(FN)} \tag{1}$$

$$f1score = \frac{2 \times precisao \times recall}{precisao + recall} \tag{2}$$

Na Figura 1 é vista a diferença do resultado das métricas de desempenho aumentando-se o desbalanceamento em um conjunto de dados. Logo, é visto que a acurácia vai aumentando, ao contrário do *recall*, mais adequado pra fazer a medição e se ter uma real dimensão do desempenho do código na base de dados.

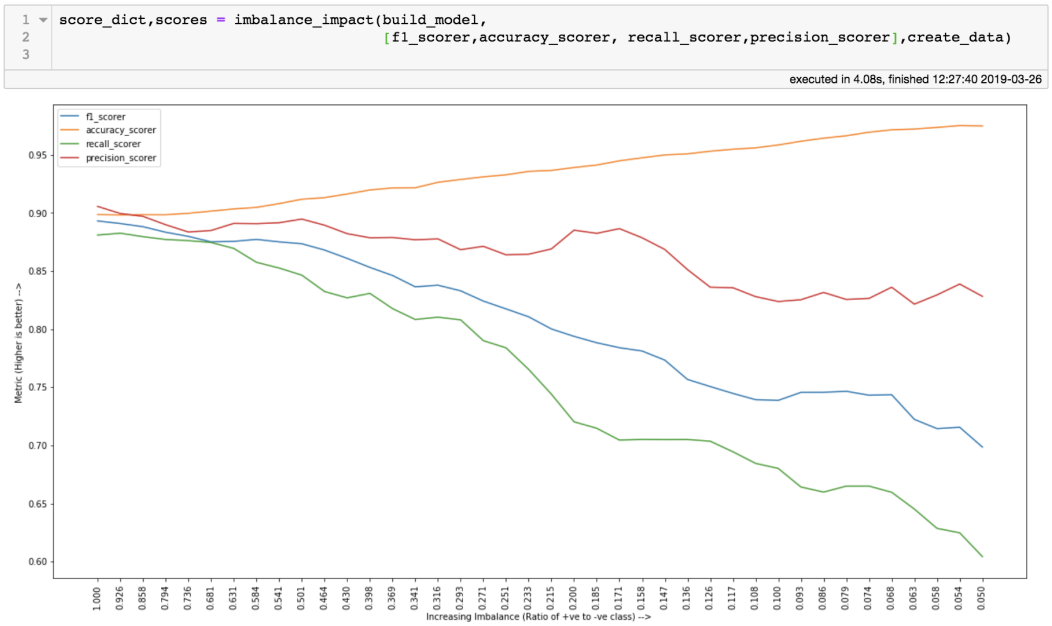

Figura 1: Gráfico da acurácia em relação ao desbalanceamento dos dados.

2 Metodologia

Foram usados os seguintes algoritmos de aprendizagem:

- Árvore de decisão;
- k-Vizinhos mais próximos;
- Redes neurais (MLP - *Multi-Layer Perceptron*).

Em cada um desses algoritmos, foram selecionadas configurações de hiper parâmetros para se testar os melhores resultados de acurácia no conjunto de treino dos dados. Para realizar a buscar da melhor configuração de cada algoritmo, foi utilizada a função *GridSearchCV()* do *sklearn*, que faz a combinação de todas as variações de parâmetros selecionados, a partir da divisão do conjunto de treino em treino e validação (*cross-validation*).

Após obtido a configuração com melhor resultado no conjunto de treino, esses parâmetros são então selecionados para serem usados no conjunto de teste, e então ter-se uma acurácia final do modelo.

3 Algoritmos de aprendizagem

3.1 Árvore de decisão

A seguir são elencados os parâmetros e suas funções no algoritmo.

- *Criterion*: aqui é definida uma função que mede a qualidade de um *split* (divisão), podendo assumir os valores *entropy* e *gini*, sendo este o *default*;
- *Splitter*: estratégia utilizada para a divisão, pode ser *random* ou *best*, sendo este último o *default*;
- *max_depth*: profundidade máxima da árvore, ou a altura, a distância do nó mais profundo até a raiz. Feito um teste inicial com as bases, foi visto que com os parâmetros *defaults* do classificador, teríamos como profundidade máxima 9. Então, foi limitamos essa altura para alguns valores para conferir a variação de acurácia, pois quanto maior e mais profunda for uma árvore, mais específica ela se torna para a base de treinamento, e o objetivo geral é a de generalização do modelo, ou seja, que seja funcional para dados ainda não vistos. Se com menos profundidade o desempenho se manter o mesmo ou bastante próximo, tende-se a optar por esse valor, para simplificar o modelo e torná-lo mais propenso à generalização;
- *random_state*: foi deixado fixo para melhor comparação dos resultados entre os modelos.

As variações de parâmetros para a árvore de decisão aplicadas no conjunto de treino do Hipertireoidismo e Hipotiroidismo foram as seguintes:

```
params = {
    'criterion': ['entropy', 'gini'],
    'splitter': ['random', 'best'],
    'max_depth': [2, 3, 4, 5, 6],
    'max_features': [15, 20, 25],
    'random_state': [10]}
```

O resultado do desempenho obtido em cada classificador pode ser visto na Figura 2.

As melhores configurações de parâmetros (e suas respectivas acurácias) foram:

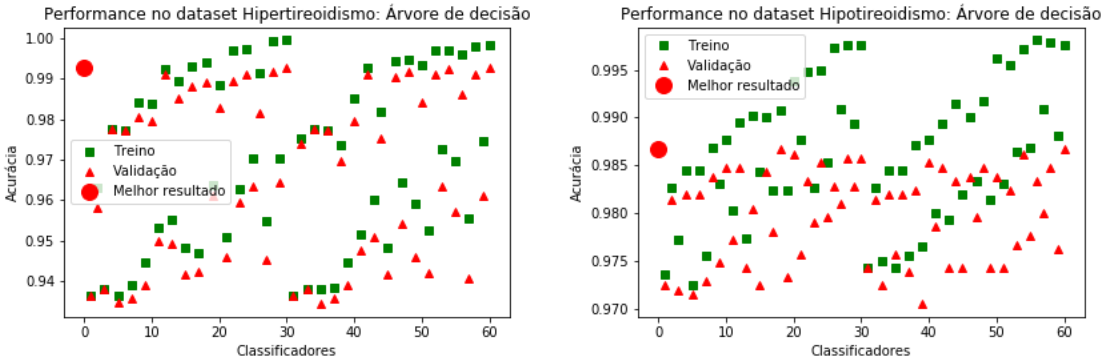

Figura 2: Acurácia obtida pelas variações de configuração na árvore de decisão.

Hipertireoidismo:
'criterion': 'entropy', 'max_depth': 6, 'max_features': 25, 'random_state': 10, 'splitter': 'best', acurácia: 0.9928571

Hipotireoidismo:
'criterion': 'entropy', 'max_depth': 4, 'max_features': 25, 'random_state': 10, 'splitter': 'best', acurácia: 0.9867

Assim, por meio da melhor configuração, foi utilizado os respectivos parâmetros no conjunto de teste:

```
Resultado no conjunto de teste (Hiper): 100.0%
Resultado no conjunto de teste (Hipo): 99.57143%
```

Na escolha das configurações, vê-se que a profundidade da árvore foi limitada no segundo caso (Hipotireoidismo). Isso significa que o aumento na altura não implicou na melhoria da divisão dos ramos. Outra implicação é uma melhor visualização dos dados e uma melhor generalização, reduzindo-se a altura, pois uma árvore muito grande significa que o modelo adaptou-se especificamente para um conjunto de dados (*overfit*). A divisão sendo a *best* já é esperada, no lugar da aleatória, e o número de *features* considerada para a divisão foi o maior número dentre as opções, o que significa que todos foram importantes para serem considerados na divisão.

3.2 k-Vizinhos mais próximos

O k-*Nearest Neighbors* ou k-Vizinhos mais próximos, é uma técnica onde não se possui processamneto na fase de treinamento. Os parâmetros variados foram o número k de vizinhos, o algoritmo usado para computar os vizinhos, a função peso usada na predição e o tipo de métrica de distância a ser comparada (potência da métrica de Minkowski).

```
params = {
    'n_neighbors': [2, 3, 4, 5, 6, 7, 8, 9, 10],
    'algorithm': ['ball_tree', 'kd_tree', 'brute'],
    'weights': ['distance', 'uniform'],
    'p': [1, 2]}
```

As melhores configurações de parâmetros (e suas respectivas acurácias) foram:

Hipertireoidismo: 'algorithm': 'ball_tree', 'n_neighbors': 3, 'p': 2, 'weights': 'distance', acurácia: 0.9495

Hipotireoidismo: 'algorithm': 'ball_tree', 'n_neighbors': 5, 'p': 1, 'weights': 'distance', acurácia: 0.979524

Na Figura 3, mostra-se o desempenho dos classificadores obtidos nas bases de dados.

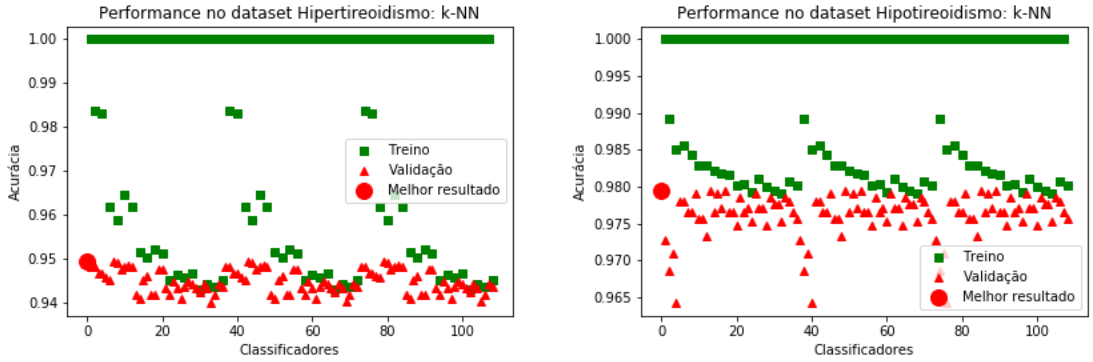

Figura 3: Acurácia obtida pelas variações de configuração no k-NN.

```
Resultado no conjunto de teste (Hiper): 100.0%
Resultado no conjunto de teste (Hipo): 100.0%
```

O algoritmo ajustou-se com valores baixos no número de vizinhos, com 3 e 5. Vale notar que se não fosse feito a divisão do treino para o conjunto de validação, simplesmente considerando-se a acurácia no conjunto de treino obteria-se acurácia maior quanto menor for o k. Nesse contexto entra a importância de um conjunto de validação para o modelo ser menos enviesado.

3.3 Redes neurais

Diferente da árvore de decisão e do k-NN, na rede neural temos maior possibilidade de variação de parâmetros, devido a complexidade da arquitetura de uma rede neural. Aqui existem várias possibilidades de formato da rede, de regularização que pode resolver problemas de *overfitting*, velocidade de aprendizado, e de forma de ativação do neurônio. Um dos parâmetros que mais influenciam a resposta é a função de ativação, e como *default* temos a *relu*. A arquitetura da rede também influcenciará bastante. Como variação de parâmetros, tem-se:

```
params = {
    'hidden_layer_sizes': [[(10)], (3, 3)],
    'activation': ['relu', 'tanh'],
    'solver': ['adam', 'lbfgs', 'sgd'],
    'alpha': [0.01, 0.1, 1, 10],
    'learning_rate_init': [0.1, 1],
    'max_iter': [200],
    'random_state': [10]}
```

As melhores configurações de parâmetros (e suas respectivas acurácias) foram:

Hipertireoidismo: 'activation': 'tanh', 'alpha': 1, 'hidden_layer_sizes': (3, 3), 'learning_rate_init': 0.1, 'max_iter': 200, 'random_state': 10, solver: lbfgs, acurácia: 0.9681

Hipotireoidismo: 'activation': 'tanh', 'alpha': 0.1, 'hidden_layer_sizes': [10], 'learning_rate_init': 0.1, 'max_iter': 200, 'random_state': 10, solver: lbfgs, acurácia: 0.98143

Finalmente, o gráfico de acurácia das configurações de classificadores pode ser vista em Figura 4.

Usando a melhor configuração no conjunto de teste obtém-se:

```
Resultado no conjunto de teste (Hiper): 95.14%
Resultado no conjunto de teste (Hipo): 99.0%
```

A justificativa para a configuração da arquitetura em uma rede neural é mais complexa, pois esta funciona como uma caixa preta. O melhor valor para o parâmetro de taxa de aprendizado é obtido por tentativa e erro. Já o *alpha*, pode-se ter alguma previsão pois ele é o que estará penalizando,

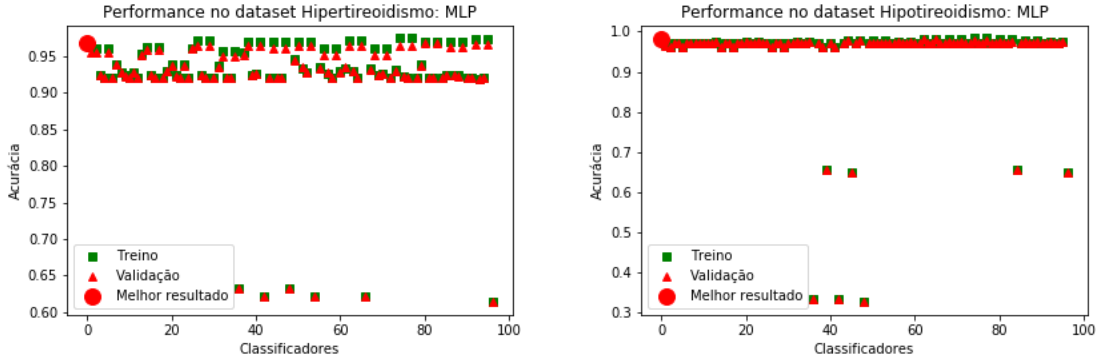

Figura 4: Acurácia obtida pelas variações de configuração na MLP.

servindo como termo de regularização. Ou seja, se o modelo está com *overfitting*, aumenta-se a regularização para suavização e melhor ajuste do modelo.

4 Conclusão

Como descrito anteriormente, a dificuldade na avaliação dos algoritmos de aprendizagem está no mal uso da métrica de acurácia para uma base de dados bastante desbalanceada. Nesse contexto, tem-se um resultado facilmente chegando próximo de 100%, mesmo com grande variação dos parâmetros.

Outra questão está na divisão do conjunto de treino para validação cruzada na função *GridSearchCV()*. Como algumas classes possuem poucas observações (a menor possui somente um), o próprio algoritmo gera um aviso de que o número de divisões é maior que o número de exemplares de alguma classe. Isso gera conjuntos de separação onde não há representação da classe, sendo portanto, impossível para o classificador prever aquela classe no conjunto de validação.