UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

INSTITUTO DE FÍSICA

RELATÓRIO DE ATIVIDADES

Aluno: Pedro Henrique Mendes
Orientador: Marco Aurélio Pires Idiart

Porto Alegre, Junho de 2021

DESENVOLVIMENTO DE MÉTODO COM ALTA ESPECIFICIDADE BASEADO EM INTELIGÊNCIA ARTIFICIAL PARA IDENTIFICAÇÃO E TRIAGEM RÁPIDA E INTELIGENTE DE CASOS DE COVID-19 USANDO EXAMES DE SANGUE SIMPLES

Resumo

Identificada no final de 2019 na China, o COVID-19 se espalhou rapidamente por todo mundo. Diversos estudos estão sendo feitos para entender melhor a doença. Entre as linhas de pesquisa esta a de estudar os biomarcadores dos casos graves de COVID-19. Os biomarcadores podem ser auxiliar a linha de frente de combate a identificar os casos graves que necessitam um atendimento imediato. O presente relatório trata do estudo da análise de biomarcadores através de Aprendizado de Máquina realizado desde Março de 2021.

Introdução

Estudos utilizando Aprendizado de Máquina, *Machine Learning* (ML), tem crescido nos últimos anos. Com uma ótima capacidade de identificar padrões nos dados, ML é utilizada para diversos fins. Uma de suas utilizações é, dado uma informação de entrada, tentar prever a saída correta. Em um conjunto de dados rotulados podemos estar interessados em prever ou classificar o resultado alvo. Esse processo é chamado de Aprendizado Supervisionado, *Supervised Learning* (sL), uma subcategoria de ML, que foi utilizado nesse estudo.

Baseado na Ref. [1] foi analisado um conjunto de dados (*dataset*) obtido no Hospital Tongji, em Wuhan, China, entre 10/01/2020 e 18/02/2020. Esse *dataset* contém 375 casos de COVID-19 onde 201 pacientes se recuperaram e 174 vieram a óbito. As informações clínicas são os rótulos utilizados no treinamento do algoritmo e o resultado da doença, alta ou óbito, o alvo que queremos prever. O principal objetivo de realizar esse estudo é identificar quais biomarcadores estão ligados aos casos mais graves de COVID-19.

Os códigos foram desenvolvidos utilizando Python 3.7 utilizando a plataforma Google Colaboratory [1]. Os códigos são de livre acesso e estão disponiveis em https://github.com/pedhmendes/covid-ml-ic.

Métodos

O *dataset* possuí muitos dados faltantes, e isso poderia prejudicar a previsão de alta/óbito dos pacientes. Assim foram utilizados como rótulos apenas os dados que possuíam menos que 20% de dados faltantes. Figura 1 mostra um gráfico em barra da porcentagem de dados faltantes. Como ainda sim possuímos dados faltantes é necessário utilizar métodos de aprendizado que consigam lidar com isso. Uma opção é utilizar os métodos baseados em árvores de decisão, então foi utilizado dois métodos: *Random Forest* (RF) e *Extreme Gradient Boosting* (XGB), ambos da biblioteca XGBoost [2].

Ambos treinamentos foram realizados da mesma forma: A taxa de aprendizado, *learning rate*, foi tomada como 0.2, a profundidade máxima (número de folhas da árvore) utilizada foi 2 e quantidade de estimadores (número de árvores) foram 150. O modelo foi executado 50 vezes, e em cada iteração foi utilizada uma semente diferente para o gerador aleatório separar os dados de treino e teste. Dados esses que eram sempre 70% do *dataset* seria usado para o treino e 30% para o teste.

[1] https://colab.research.google.com/
[2] https://xgboost.readthedocs.io/en/latest/

Os algoritmos que utilizamos são de uma classe chamada de algoritmos de *ensembles* (conjuntos). Esses algoritmos possuem diversas árvores de decisão, os números de estimadores citados anteriormente, e ao final realizam uma votação, média, para prever o resultado. Nesses processos os rótulos terão pesos nas votações e é exatamente isso que mediremos. A cada iteração do treinamento iremos guardar os pesos dos rótulos, depois escolheremos apenas os pesos que possuem mais importância para a previsão e esses serão utilizados para o modelo.

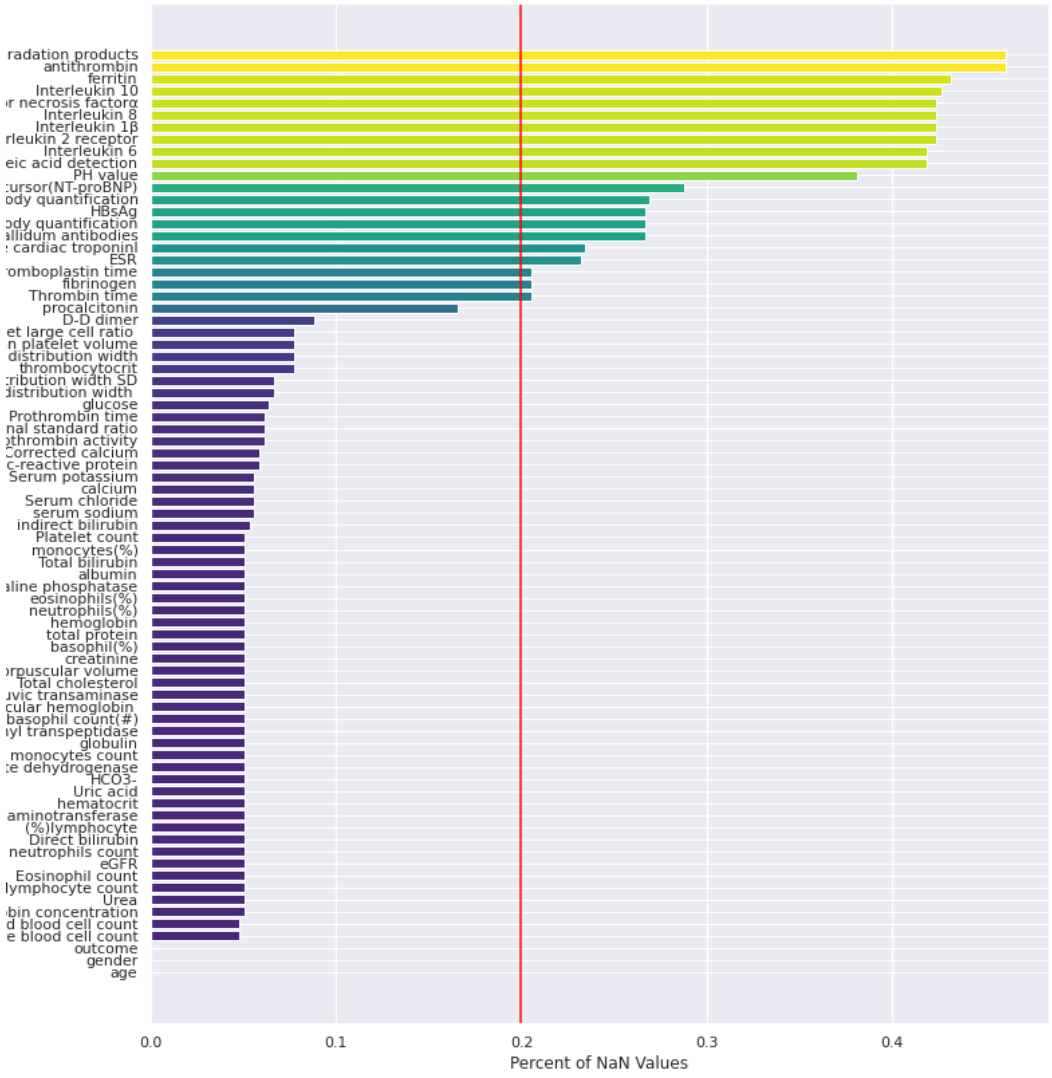

Figura 1: Porcentagem de dados faltantes para cada informação de rótulo e linha marcando os 20%.

Após eliminar os dados que tinham mais de 20% de dados faltantes ficamos com 55 rótulos disponíveis, de um inicial de 76. Também foram eliminados as datas de admissão e alta/óbito, pois essa informação não nos interessa.

Desenvolvimento

Primeiramente treinaremos o algoritmo, *Random Forest* ou *Extreme Gradient Boosting*, 50 vezes coletando os pesos de cada rótulo. Veja na Figura 2 um trecho do código utilizado.

```
[12] import_feature_RF = pd.DataFrame(columns=X.columns)

     tmax = 50 #100, 150, 200

     for i in range(tmax):
         X_train, X_test, y_train, y_test = train_test_split(X,
                                                             y,
                                                             test_size=0.3,
                                                             random_state=i) #state for iteration

         model_RF = XGBRFClassifier(max_depth=4,             #Maximum tree depth for base learners.
                                    learning_rate=0.2,       #learning rate ("eta")
                                    reg_lambda=1,            #L2 regularization term on weights
                                    n_estimators=150,        #Number of boosting rounds.
                                    subsample = 0.9,         #Subsample ratio of the training instance.
                                    colsample_bytree = 0.9)  #Subsample ratio of columns when constructing each tree.

         model_RF.fit(X_train, y_train)
         import_feature_RF = import_feature_RF.append(pd.DataFrame(model_RF.feature_importances_,
                                                                   index=X.columns).transpose())
```

Figura 2: Trecho do treinamento e a forma utilizada para guardar os pesos importantes dos métodos.

Então ordenamos os dados os dados por ordem de importância. Com isso podemos selecionar os cinco dados que tem mais peso para a previsão do alvo. Veja na Figura 3 os dados mais importantes, à esquerda para quando utilizamos RF e à direita quando utilizamos XGB.

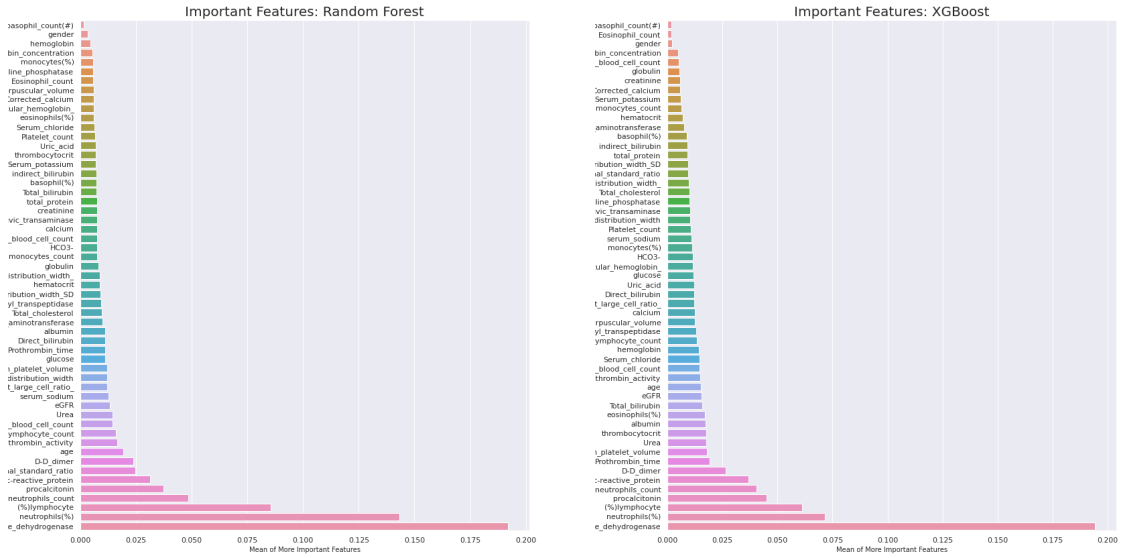

Figura 3: (*Esq*) Rótulos ordenamos por ordem de importância utilizando RF. (*Dir*) Rótulos ordenamos por ordem de importância utilizando XGB.

É possível ver que, utilizando RF os rótulos mais importantes são *Lactate dehydrogenase*, *neutrophils(%)*, *(%)lymphocyte*, *neutrophils count* e *procalcitonin*, nessa ordem. Utilizando XGB obtemos a seguinte ordem de importância: *Lactate dehydrogenase*, *neutrophils(%)*, *(%)lymphocyte*, *procalcitonin* e *neutrophils count*. Obtemos os mesmos cinco rótulos, porém os dois últimos em ordem invertida. Após definido os rótulos mais importantes vamos avaliar o nosso modelo. Na Figura 4 encontra um trecho do código utilizado para avaliar o modelo.

Após avaliar o modelo podemos montar a matriz de confusão, responsável por mostrar a qualidade do modelo. As matrizes de confusão para ambos métodos utilizados estão dispostas na Figura 5 A diagonal principal da matriz de confusão mostra os acertos do modelo e a diagonal secundaria mostra os erros. Pela barra de cor ao lado, podemos ver que nosso modelo tem um número considerável de acertos, mas apenas isso não é suficiente.

```
[15] X_train, X_test, y_train, y_test = train_test_split(X_best_RF, #x = x_best
                                                         y, #same
                                                         test_size=0.3, #same
                                                         random_state=3463) #def state

    model_RF = XGBRFClassifier(max_depth=4,
                               learning_rate=0.2,
                               reg_lambda=1,
                               n_estimators=150,
                               subsample=0.9,
                               colsample_bytree=0.9,
                               verbosity=0)

    model_RF.fit(X_train,y_train)
```

Figura 4: Trecho da parte do código responsável por avaliar a qualidade do modelo.

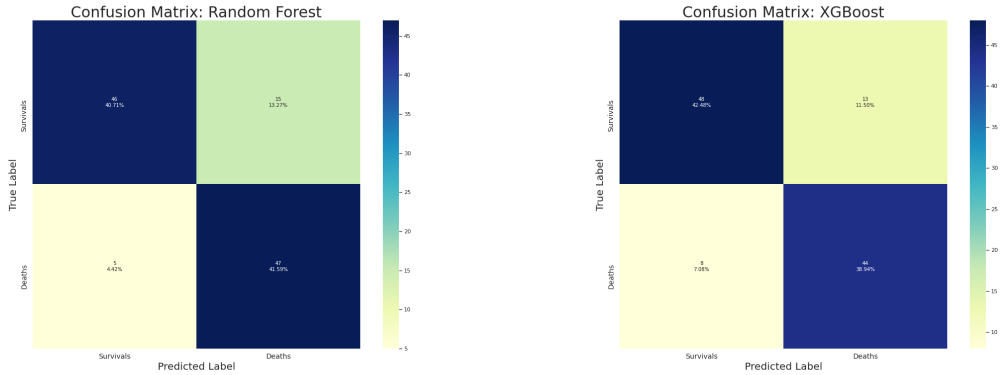

Figura 5: (*Esq*) Matriz de confusão utilizando RF. (*Dir*) Matriz de confusão utilizando XGB.

Das matrizes de confusão obtemos que os F1-Score [3], que é responsável por avaliar a qualidade do modelo. Obtemos 0.82(4) para o algoritmo RF e 0.80(7) para o algoritmo XGB. O último algarismo é a incerteza do algoritmo.

Resultados e Perspectivas

Utilizando ambos algoritmos, RF e XGB, obtemos que os biomarcadores mais importantes são *Lactate dehydrogenase*, *neutrophils(%)*, *(%)lymphocyte*, *neutrophils count* e *procalcitonin*, entretanto algumas ressalvas precisam ser feitas. Os F1-Scores obtidos foram de 0.82(4) para RF e 0.80(7) para XGB, quando treinados com os hiperparâmetros citados e utilizando 50 iterações.

O estudo foi feito com apenas 375 pacientes e 77 rótulos, onde apenas 55 rótulos foram utilizados. Possuímos muitos dados faltantes e um número pequeno de pacientes, assim o resultado pode ser especifico para esse conjunto de pacientes. Dada a importância da doença é de se esperar que mais dados estejam disponíveis, mas isso não é observado. O número pequeno de dados junto com a grande quantidade de dados faltantes prejudicam o aprendizado da máquina, ainda que utilizemos uma biblioteca que contorna esse problema.

Futuramente será realizado uma análise estatística dos dados, levando em conta os métodos utilizados na literatura. Também é importante citar que o COVID-19 ainda esta sendo estudado, novos dados podem estar disponíveis e serem utilizados para o este proposito.

Referências

[1] L. Yan, H.-T. Zhang, J. Goncalves, Y. Xiao, M. Wang, Y. Guo, C. Sun, X. Tang, L. Jing, M. Zhang, X. Huang, Y. Xiao, H. Cao, Y. Chen, T. Ren, F. Wang, Y. Xiao, S. Huang, X. Tan, N. Huang, B. Jiao, C. Cheng, Y. Zhang, A. Luo, L. Mombaerts, J. Jin, Z. Cao, S. Li, H. Xu, and Y. Yuan, "An interpretable mortality prediction model for covid-19 patients," *Nature Machine Intelligence*, vol. 2, pp. 283–288, May 2020.

[3] https://en.wikipedia.org/wiki/F-score