A COVID-19 Study Through Machine Learning

Pedro Henrique Mendes
Advisor: Marco Aurelio Pires Idiart

Instituto de Física
Universidade Federal do Rio Grande do Sul

2021

Introduction

- Part 1: Ensemble Algorithms
- Part 2: Actual Work

Decision Trees

A decision tree is a decision support tool that uses a tree-like model.

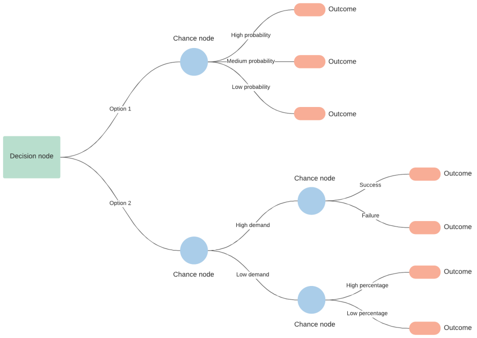


Figure 1: Decision Tree chart.

Bootstrapping

Statistical technique consisting in generate samples of size B from an initial dataset of size N by randomly drawing with replacement B observations.

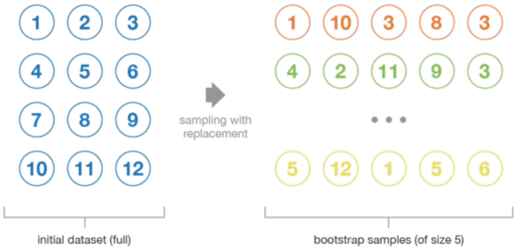


Figure 2: Bootstrapping process.

Bagging

Bagging consists in fitting several base models on different bootstrap samples and build an ensemble model that “average” the results of these weak learners.

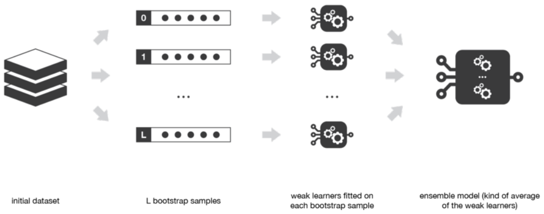

0
1
...
L
...
initial dataset
L bootstrap samples
weak learners fitted on each bootstrap sample
ensemble model (kind of average of the weak learners)

Figure 3: Bagging process

Random Forest

The random forest approach is a bagging method where deep trees, fitted on bootstrap samples, are combined to produce an output with lower variance.

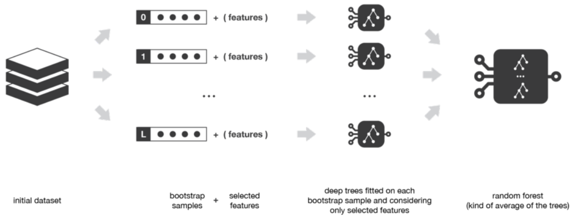

0
+ (features)
1
+ (features)
...
L
+ (features)
...
initial dataset
bootstrap samples
+
selected features
deep trees fitted on each bootstrap sample and considering only selected features
random forest (kind of average of the trees)

Figure 4: Random Forest process

Data

Medical information of collected between 10 January and 18 February 2020, from Tongji Hospital. Were excluded the data from patients with more than 80% of missing data.

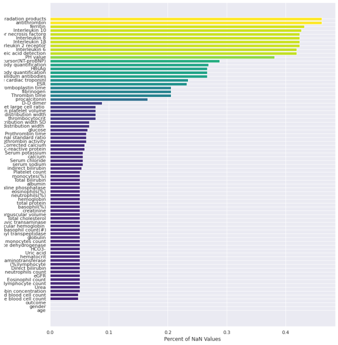
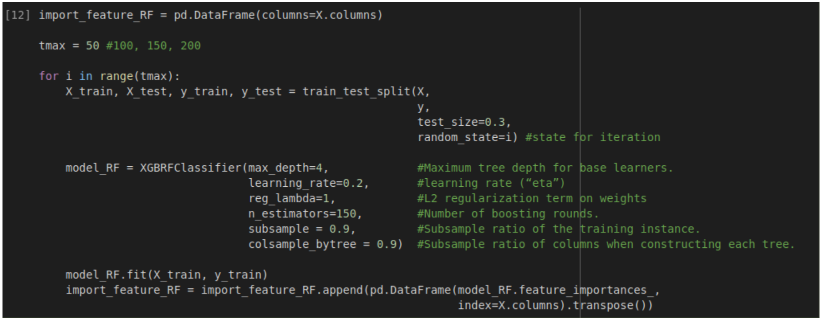


Figure 5: Missing data.

Training

Using XGBoost's Random Forest Classifier we train our model.



```
[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())
```

Figure 6: Training

Best Features

We can see which feature has more weight.

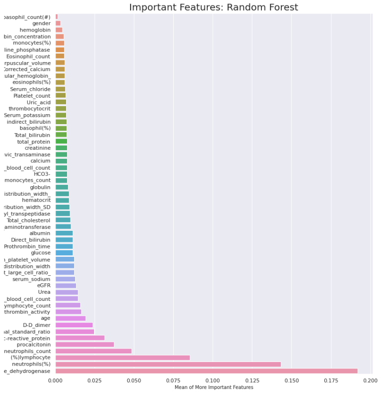
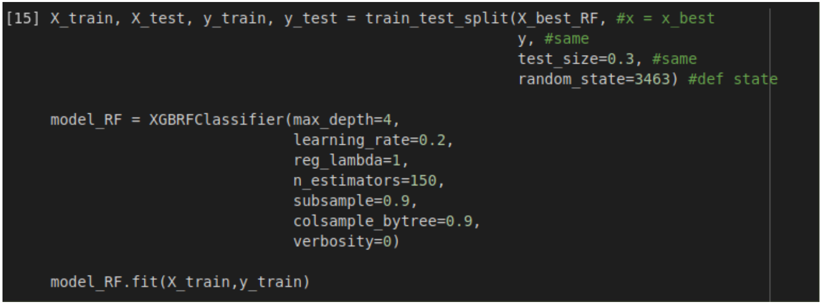


Figure 7: Important Features

Evaluation

According to Figure (7) our relevant features are: Lactate Dehydrogenase, Neutrophils percent, Lymphocyte percent, Neutrophils Count and Procalcitonin. We then fit a model using only this features.



```
[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)
```

Figure 8: Evaluating our model

Results

We can see our results better in a confusion matrix.

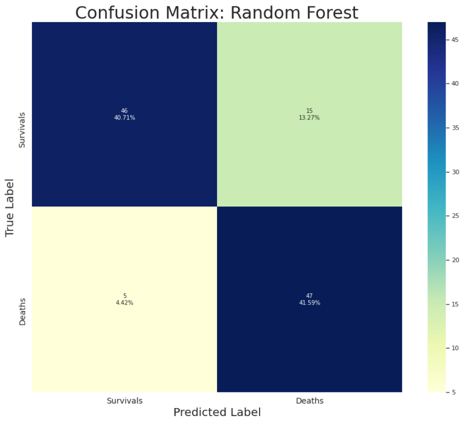
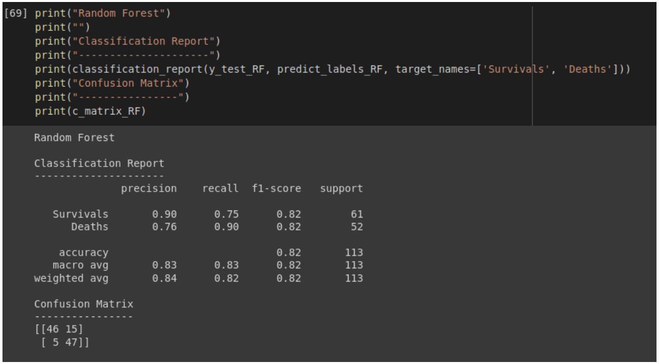


Figure 9: Confusion Matrix

Results

And retrieving our Classification Report we can see that our model has a confiability of 82%.



```
[69] print("Random Forest")
     print("")
     print("Classification Report")
     print("---------------------")
     print(classification_report(y_test_RF, predict_labels_RF, target_names=['Survivals', 'Deaths']))
     print("Confusion Matrix")
     print("----------------")
     print(c_matrix_RF)
```

```
Random Forest

Classification Report
---------------------
              precision    recall  f1-score   support

   Survivals       0.90      0.75      0.82        61
      Deaths       0.76      0.90      0.82        52

    accuracy                           0.82       113
   macro avg       0.83      0.83      0.82       113
weighted avg       0.84      0.82      0.82       113

Confusion Matrix
----------------
[[46 15]
 [ 5 47]]
```

Figure 10: Classificarion Report

Perspectives

- Learn how XGBoost algorithms handle missing data
- Create a better understanding of the important biomarkers.
- Adapt our model to Gradient Boosted Trees and reproduce the results of Yang et al. [1]

[1] Yang et al, "*An interpretable mortality prediction model for COVID-19 patients*", 2020

References

- Figure (1): https://lucidspark.com/blog/how-to-make-a-decision-tree. Accessed on May 2021.
- Figures (2), (3) and (4): https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205. Accessed on May 2021.
- All codes and data are available in my github: https://github.com/pedhmendes

Acknowledgments

Thanks for coming to my TED talk.