**Naive Bayes**

Para textos **não processados**, obtivemos os seguintes valores:

-----------------------------------------------------

              test\_size = 0.15

              precision    recall  f1-score   support

           0       0.77      0.90      0.83       583

           1       0.66      0.43      0.52       268

    accuracy                           0.75       851

   macro avg       0.72      0.66      0.68       851

weighted avg       0.74      0.75      0.73       851

-----------------------------------------------------

              test\_size = 0.20

              precision    recall  f1-score   support

           0       0.78      0.89      0.83       776

           1       0.66      0.45      0.54       358

    accuracy                           0.75      1134

   macro avg       0.72      0.67      0.68      1134

weighted avg       0.74      0.75      0.74      1134

-----------------------------------------------------

              test\_size = 0.25

              precision    recall  f1-score   support

           0       0.78      0.89      0.83       971

           1       0.66      0.44      0.53       447

    accuracy                           0.75      1418

   macro avg       0.72      0.67      0.68      1418

weighted avg       0.74      0.75      0.73      1418

-----------------------------------------------------

              test\_size = 0.30

              precision    recall  f1-score   support

           0       0.77      0.89      0.83      1165

           1       0.64      0.40      0.50       536

    accuracy                           0.74      1701

   macro avg       0.70      0.65      0.66      1701

weighted avg       0.73      0.74      0.72      1701

Para **textos processados**, obtivemos os seguintes valores:

-----------------------------------------------------

              test\_size = 0.15

              precision    recall  f1-score   support

           0       0.78      0.83      0.80       583

           1       0.57      0.51      0.54       268

    accuracy                           0.73       851

   macro avg       0.68      0.67      0.67       851

weighted avg       0.72      0.73      0.72       851

----------------------------------------------------

              test\_size = 0.20

              precision    recall  f1-score   support

           0       0.80      0.83      0.81       776

           1       0.59      0.55      0.57       358

    accuracy                           0.74      1134

   macro avg       0.70      0.69      0.69      1134

weighted avg       0.74      0.74      0.74      1134

------------------------------------------------

              test\_size = 0.25

              precision    recall  f1-score   support

           0       0.79      0.82      0.81       971

           1       0.58      0.53      0.56       447

    accuracy                           0.73      1418

   macro avg       0.69      0.68      0.68      1418

weighted avg       0.73      0.73      0.73      1418

------------------------------------------------

              test\_size = 0.30

              precision    recall  f1-score   support

           0       0.80      0.82      0.81      1165

           1       0.59      0.54      0.56       536

    accuracy                           0.73      1701

   macro avg       0.69      0.68      0.69      1701

weighted avg       0.73      0.73      0.73      1701

**Random Forest Classifier**

Para textos processados e

para textos não processados

A partir dos valores

n\_estimators = [10,20,30,40,50,60,70,80,90,100,110,120,130,140,150]

max\_depth = [5, 10, None]

min\_samples\_split = [5, 10, 15, 20]

min\_samples\_leaf = [1, 2, 3, 5, 10, 20]

testamos 5 combinações aleatórias para cada *test\_size*

test\_sizes = [0.15, 0.20, 0.25, 0.30]

e, para cada combinação, selecionamos aleatoriamente

parameters = {

   'max\_features': ('auto','sqrt'),

    'n\_estimators': random.sample(n\_estimators, 3),

    'max\_depth': random.sample(max\_depth, 2),

    'min\_samples\_split': random.sample(min\_samples\_split, 2),

    'min\_samples\_leaf': random.sample(min\_samples\_leaf, 2),

    'bootstrap': [True, False]

}

Formando assim *test\_size* \* 5 = 20 combinações x 2 (processados e não processados) = 40