



TSG IT Systems

project report

**Improvement of quality of Prediction of Mechanical Ventilation by  
Decreasing its predictive False Positives**

Date	Version	Author	Validation status	Checker
08.08.2020	1.0	Galina Blokh	Not validated	L. Schvartser



TSG IT Systems

# 1.Introduction

**Noninvasive ventilation (NIV)** may be used in patients suffering from acute respiratory failure. However, failure to deliver the appropriate ventilator support leads to mechanical ventilation which is associated with higher mortality, compared to the patients ventilated from the acute onset. **Predicting the success of NIV** has been a topic of interest. (1)

Various parameters have been found to predict failure on NIV in various settings. These include disease etiologies and mainly varied physiologic markers, including markers for oxygenation and respiratory load. It should be noted that NIV failure may occur shortly after its onset or following hours and even days. Delayed recognition of that NIV failure may lead to increased morbidity and mortality. NIV and High flow Nasal Cannula Oxygenation are used in around 15% of the COVID-19 respiratory failure patients. **Our goal was to improve the prediction of NIV failure** using a newly devised model based on an existing large dataset in respiratory failure (1).

## 2.Data

A model was designed to predict the onset of invasive mechanical ventilation to answer the question: **will the patient, currently not invasively ventilated, require invasive ventilation within next 6, 12, 18 or 24 hours**, the MIMICExtract data set and pipeline for data preparation were used. The construction of an algorithm to predict mechanical ventilation **was made, using XGBoost** algorithm on time series of 6 hours with addition of static variables to every series. MIMIC Extract provides standardized data processing functions, including unit conversion, outlier detection (1).

The data included all the admissions that received invasive mechanical ventilation. The data was balanced with the admissions that didn't pass the Mechanical Ventilation such as those passed will be 39% of all the cohort. The data set for experiment is done on the 6 hours sliding intervals such that in this 6 hours and in the next 6 (or 12, 18, 24) hours after it (gap) the ventilation was not performed, so the ventilation Onset was detected within 4 hours after the gap (positives) or not detected in the same 4 hours (negatives) (1). The **data set was stratified on 'intervention'** features into training, testing and validation sets in proportion **Training set 41%, Validation set 11% and Test set 48% (3)**.

The data vector  $x_t$  for every sliding window is formed as concatenation of 6 values, value per hour for all the dynamic features plus static features (age, weight, ...) added to the same vector. For example, let us have 100 dynamic measurements and 5 static measurements. Our vector became of dimension  $100 \times 6 + 5 = 605$ . The corresponding label  $y_t$  is 1 if the Onset detected in 4 hours after the gap and 0 otherwise. All 3 indicators for every feature were in use with its value, time from last measure that reflects its accuracy, and its mask that reflects its reality. In feature importance calculation, all three importance are summarized. This approach is accepted for

medical applications where the features are measured with different rates and the imputer is used for data synchronization (1).

For our tasks we used a set **with size 100** patients data **to check the correctness** of implemented methods and a set **with size 3000** patients data, **to evaluate** new model **results**. This model is acceptable for use on the Sheba Data set.

The code was added to path **MIMIC\_Extract\_tsg/src/Vent/\_OnSet\_Sheba\_FalseP\_detection**

## 3.Methods & Results

To improve existing results which was got by evaluating on Test set, we implemented the following:

### 3.1 After the model was built on the Train set, run predict\_proba on Validation set:

For this task was changed existing function **run\_only\_final()** and **run\_basic()** which was making calculations predict\_proba() only for the Test set. There was added functionality to make predictions and save into csv file probabilities score on the Validation set. Return only saved document path.

The document path is:

**MIMIC\_Extract\_tsg/src/Vent/\_OnSet\_Sheba\_FalseP\_detection/\_Baselines\_Intervention\_Prediction\_Mechanical\_Ventilation.py**

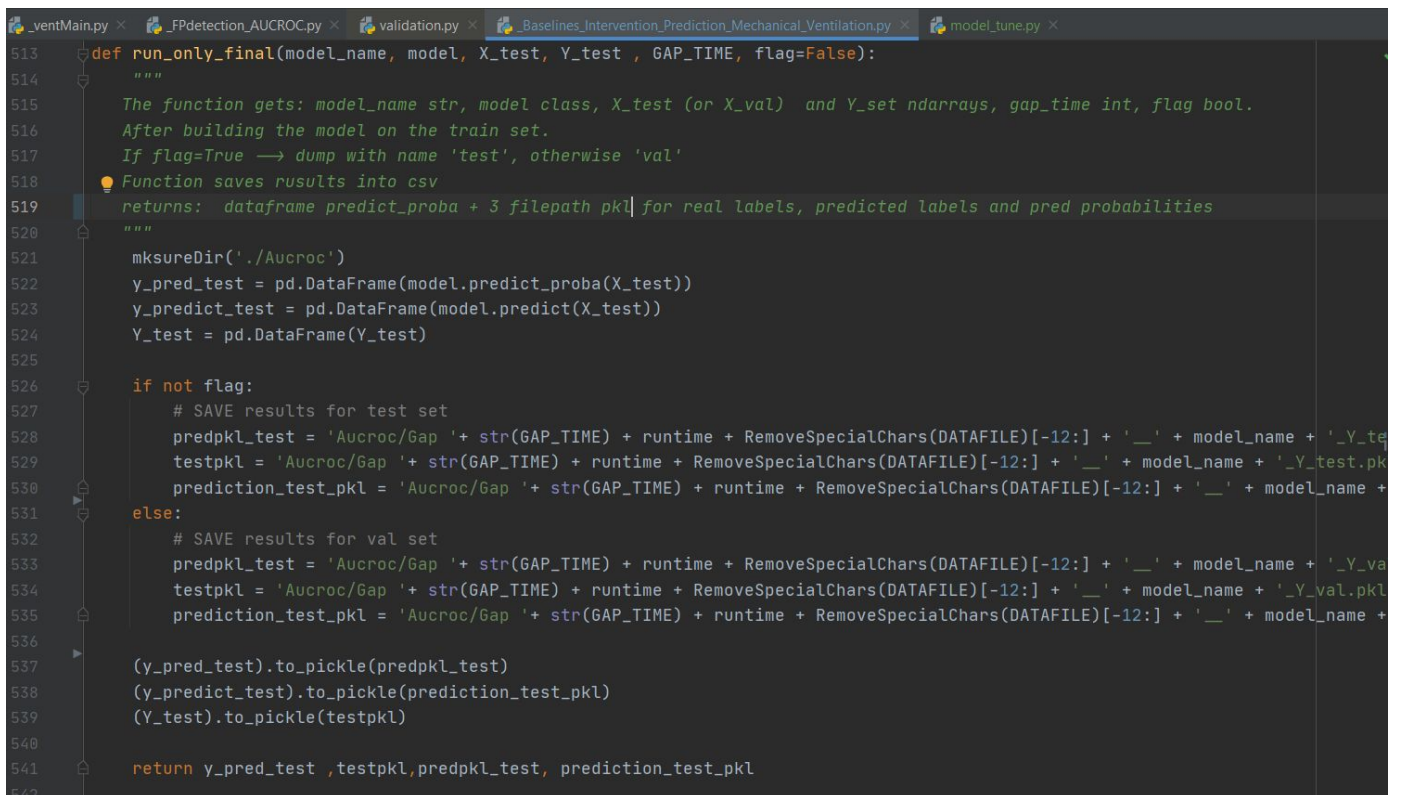


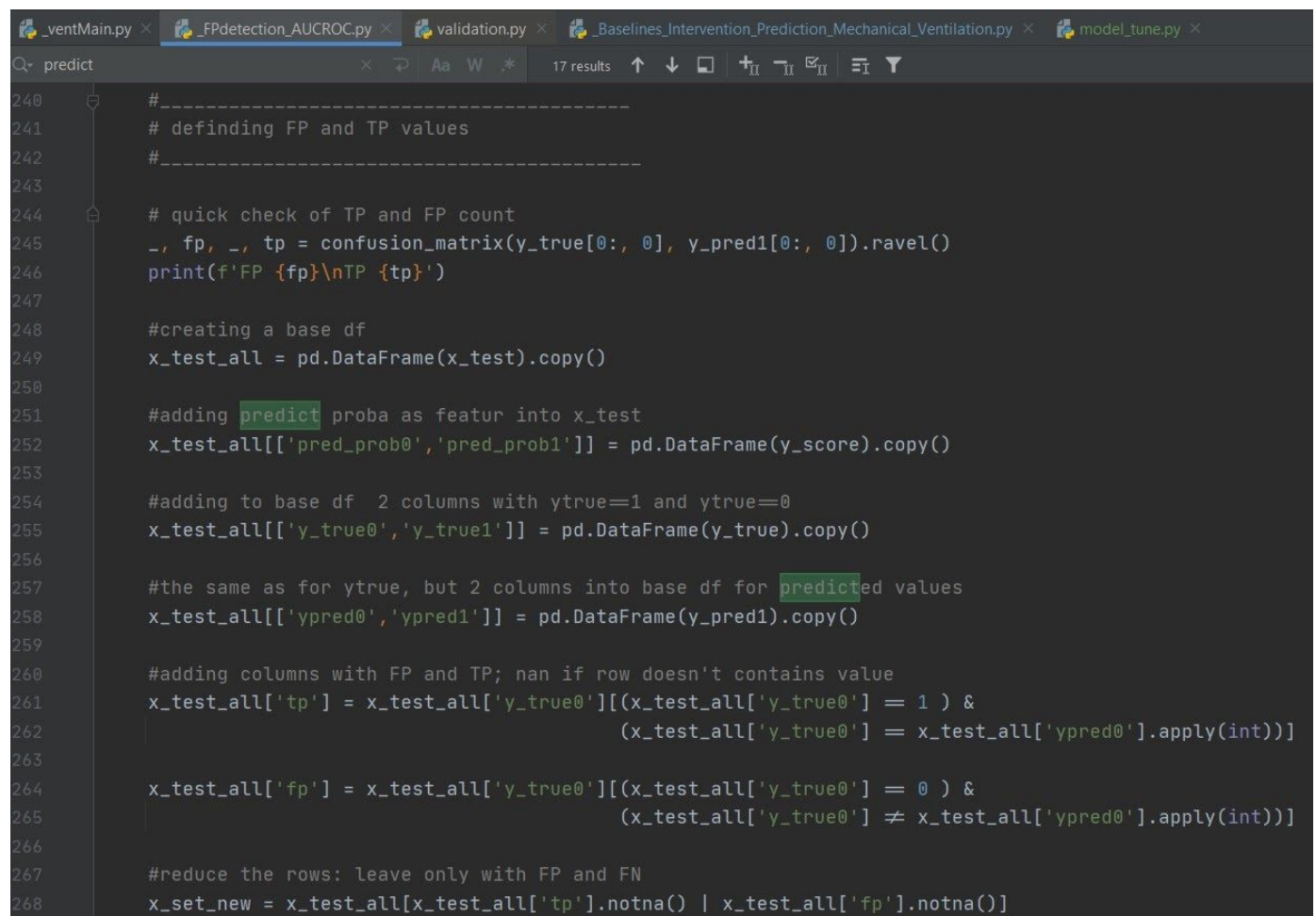
Fig.1 - Function which runs prdict\_proba and predict on Validation/Test set and saves scores into pkl file (3)

### 3.2 To find the false positives and the true positives using the threshold calculated on ROC curve:

Here we were using an already existing function **Gntbok1()** which calculates the ROC-AUC curve, confusion matrix and best threshold for the highest ROC-AUC score and other metrics for evaluating the model on the Test set.

We added functionality for calculations on the Validation set and added code to define True positives and False positives values. The path of the document is:

**MIMIC\_Extract\_tsg/src/Vent/utlis/\_FPdetection\_AUCROC.py**

A screenshot of a code editor with multiple tabs open: \_ventMain.py, \_FPdetection\_AUCROC.py (active), validation.py, \_Baselines\_Intervention\_Prediction\_Mechanical\_Ventilation.py, and model\_tune.py. The active tab shows Python code for calculating False Positives (FP) and True Positives (TP). The code includes comments and uses pandas DataFrames to process y\_true and y\_pred1 arrays. It calculates FP and TP counts, creates a base DataFrame, adds predicted probabilities, and then filters the DataFrame to keep only rows with TP or FP values. The final line creates a new DataFrame x\_set\_new based on the filtered results.

```
240 #-----
241 # defining FP and TP values
242 #-----
243
244 # quick check of TP and FP count
245 _, fp, _, tp = confusion_matrix(y_true[0:, 0], y_pred1[0:, 0]).ravel()
246 print(f'FP {fp}\nTP {tp}')
247
248 #creating a base df
249 x_test_all = pd.DataFrame(x_test).copy()
250
251 #adding predict proba as featur into x_test
252 x_test_all[['pred_prob0', 'pred_prob1']] = pd.DataFrame(y_score).copy()
253
254 #adding to base df 2 columns with ytrue==1 and ytrue==0
255 x_test_all[['y_true0', 'y_true1']] = pd.DataFrame(y_true).copy()
256
257 #the same as for ytrue, but 2 columns into base df for predicted values
258 x_test_all[['ypred0', 'ypred1']] = pd.DataFrame(y_pred1).copy()
259
260 #adding columns with FP and TP; nan if row doesn't contains value
261 x_test_all['tp'] = x_test_all['y_true0'][(x_test_all['y_true0'] == 1) &
262                                         (x_test_all['y_true0'] == x_test_all['ypred0'].apply(int))]
263
264 x_test_all['fp'] = x_test_all['y_true0'][(x_test_all['y_true0'] == 0) &
265                                         (x_test_all['y_true0'] != x_test_all['ypred0'].apply(int))]
266
267 #reduce the rows: leave only with FP and FN
268 x_set_new = x_test_all[x_test_all['tp'].notna() | x_test_all['fp'].notna()]
```

Fig.2 - Functionality to defin TP and FP values and to filter Validation set values accordingly (3)

On **Validation set** with **size 3000** patients set we got (3):

Metric	XGBoost 1 with threshold	XGBoost 1
<b>Best threshold</b>	0.0057	-
<b>ROC AUC score</b>	0.796	0.853

<b>Recall score</b>	0.792	0.922
<b>Precision score</b>	0.0335	0.0081
<b>Accuracy score</b>	0.7999	0.0084
<b>ROC AUC score macro</b>	0.796	0.853
<b>FP count</b>	2337	-
<b>TP count</b>	73	-

Table 1. *Evaluation metrics for XGBoost 1 on Validation set (before running XGBoost 2)(3)*

Here we can see confusion matrix and AUC-Roc curve for **XGBoost 1 on Validation set**:

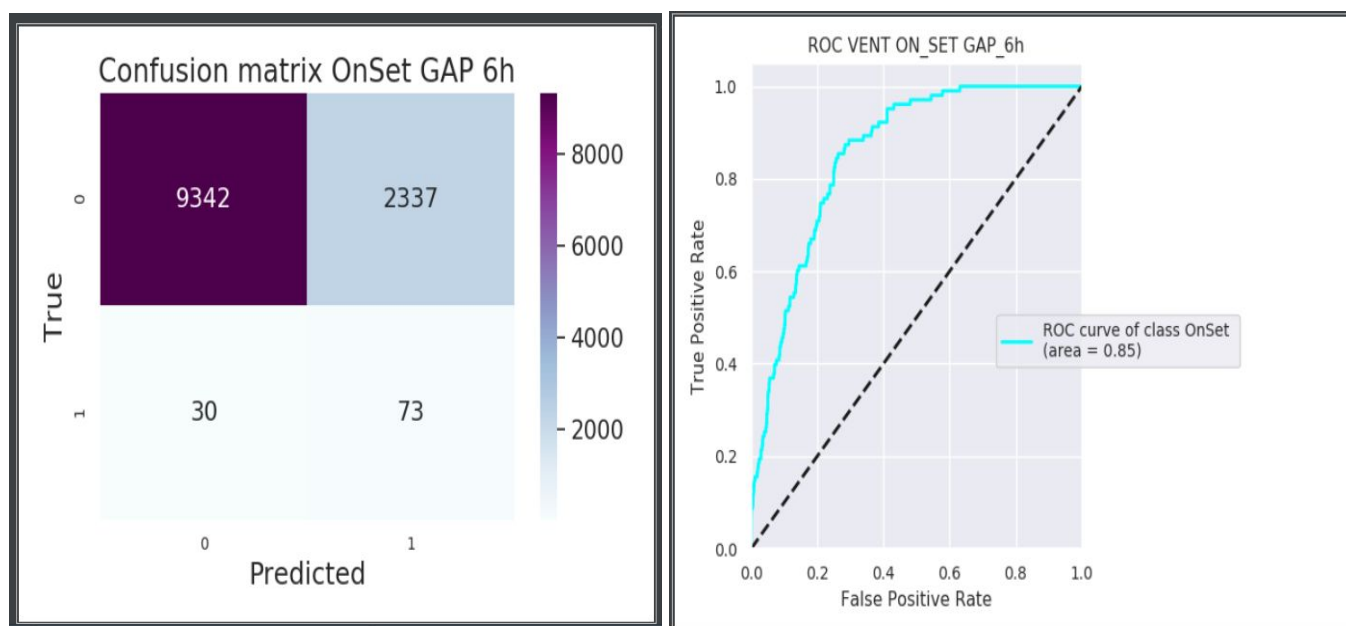


Fig.3 - *Confusion matrix and ROC-AUC curve for Validation set before run second model (3)*

### 3.3 To assign true positives as "true" and false positives as "false":

After we got first results, we added more functionality into the same **Gntbok1()** function, to assign true positives as "1" and false positives as "0", by the way added return statement with new validation x matrix and y vector, which contains only TP and FP values (3):

```
267
270     # create variables to return
271     x_test_new = x_set_new.iloc[:,0:-6].copy().to_numpy()
272
273     #labeling TP as 1, FP as 0
274     x_set_new.fp.fillna(1, inplace=True)
275     #making new y 1d array
276     y_test_new = 1 - x_set_new.fp.copy().to_numpy()
```

Fig.4 - Labeling TP as 1, FP as 0, creating new Validation X and Y only with TP and FP values to run the second model (3)

### 3.4 To run XGBoost only on the chosen set:

After all manipulations we fitted a second **XGBoost** model **on a new Validation set with only TP and FP values**. We were using the same format as X test has - numpy array.

Also we added **predict\_proba** as a feature into X-test and X-val sets and added class ballance hyperparameter **scale\_pos\_weight** for XGBoost 2:

```
642 | spw = sqrt(len(y_val_ftp[y_val_ftp ==0])/len(y_val_ftp[y_val_ftp ==1]))
643
644     #definding a new xgboost model
645     xgb2 = XGBClassifier(scale_pos_weight=spw)
646
647     y_test_true = pd.read_pickle(testpkl).to_numpy()
648     # fitting on new validation set
649     xgb2.fit(x_val_ftp, y_val_ftp)
650
651     # making prediction on test set with 2nd xgb model
652     test_prd_xgb2 = pd.DataFrame(xgb2.predict_proba(np.nan_to_num(x_test_new)))
653     test_predict_xgb2 = pd.DataFrame(xgb2.predict(np.nan_to_num(x_test_new)))
654
655     test_prd_xgb2_pkl = 'Aucroc/Gap 6 ' + runtime + RemoveSpecialChars(DATAFILE)[-12:] + '__test_pred_xgb2.pkl'
656     test_prd_xgb2.to_pickle(test_prd_xgb2_pkl)
657     predict_xgb2 = 'Aucroc/Gap 6 ' + runtime + RemoveSpecialChars(DATAFILE)[-12:] + '__prediction_xgb2.pkl'
658     test_predict_xgb2.to_pickle(predict_xgb2)
659     # evaluating 2nd xgboost model
660     x_val_xgb2, y_val_xgb2, _ = Gntbok1(x_test_new, test_prd_xgb2_pkl, testpkl, predict_xgb2, 6)
661
```

Fig.5 Run XGBoost only on the chosen set with TP and FP (3)

### 3.5 To calculate AUC, confusion matrix on this new classification

To do this part we called again the function **Gntbok1()**. But results we got was surprising:

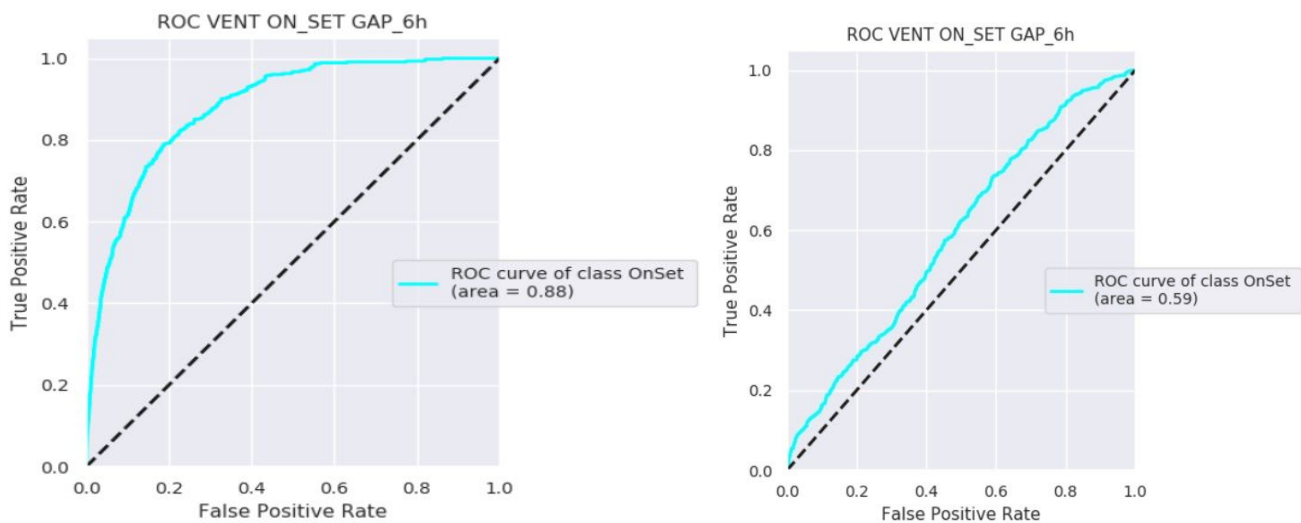


Fig.6 ROC- AUC on Test set: left - before 2nd XGBoost, right - after 2nd XGBoost (3)

Let's look to the Confusion matrices:

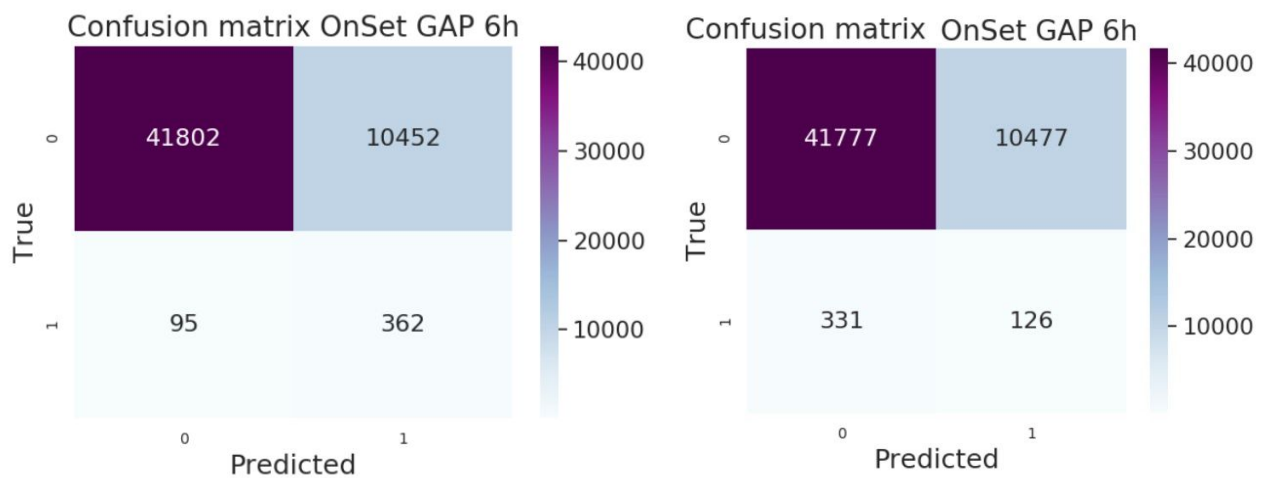


Fig.7 Confusion matrices for X Test set: left - before second XGBoost, right - after second XGBoost (3)

This is definitely not the result which we were expecting. The model doesn't work well yet.

We can look in Table 2 - the comparison of evaluating metrics for both models:

<b>Model Metric</b>	<b>XGBoost 1 with Threshold</b>	<b>XGBoost 1 real</b>	<b>XGBoost 2 with Threshold</b>	<b>XGBoost 2 real</b>
<b>Best threshold</b>	0.0057	-	0.057	-



<b>ROC AUC score</b>	0.796	0.88	0.538	0.5888
<b>Recall score</b>	0.792	0.958	0.276	0.921
<b>Precision score</b>	0.0335	0.0083	0.0119	0.0082
<b>Accuracy score</b>	0.7999	0.00873	0.795	0.0325
<b>ROC AUC score macro</b>	0.796	0.88	0.538	0.5888
<b>FP count</b>	10452	-	10477	-
<b>TP count</b>	362	-	126	-

Table 2. Evaluation metrics for 2 models with and without threshold(3)

Accuracy: 78.71% ( with std 19.45%)

Definitely we have to improve the model.

These results gave us nothing good yet in our project goal.

## 4.Improving the model

We added into the Test set and Validation set predict proba values as two columns, but it didn't give us better results.

Then we decided to tune the model with python tools for hyperparameter tuning.

### 4.1 HYPEROPT:

It is a powerful python library that searches through an hyperparameter space of values . Existing since 2012-2013 year. It implements three functions for minimizing the cost function:

- 1.Random Search
- 2.TPE (Tree Parzen Estimators)
- 3.Adaptive TPE

But in our implementation there appeared some errors which lead into the source code of the library, thus we refused to use this tool/library and went to classic tools, such as GridSearch and RandomSearch.

### 4.2 GridSearch:

The first try was running GridSearch with only 3 parameters: learning\_rate, n\_estimators and max\_depth. Here the results:



```

Run the second model
Fitting 10 folds for each of 96 candidates, totalling 960 fits
[Parallel(n_jobs=10)]: Using backend LokyBackend with 10 concurrent workers.
[Parallel(n_jobs=10)]: Done 30 tasks      | elapsed: 13.2s
[Parallel(n_jobs=10)]: Done 180 tasks    | elapsed: 2.0min
[Parallel(n_jobs=10)]: Done 430 tasks    | elapsed: 5.3min
[Parallel(n_jobs=10)]: Done 780 tasks    | elapsed: 11.3min
[Parallel(n_jobs=10)]: Done 960 out of 960 | elapsed: 14.5min finished
XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
               colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
               importance_type='gain', interaction_constraints=None,
               learning_rate=0.1, max_delta_step=0, max_depth=4,
               min_child_weight=1, missing=nan, monotone_constraints=None,
               n_estimators=140, n_jobs=0, num_parallel_tree=1,
               objective='binary:logistic', random_state=34000, reg_alpha=0,
               reg_lambda=1, scale_pos_weight=0.1767388699628784, seed=34000,
               subsample=1, tree_method=None, validate_parameters=False,
               verbosity=None)

```

*Fig.8 GridSearchCV results best\_estimator\_for second XGBoost ( learning\_rate, n\_estimators and max\_depth)(3)*

Then we decided to run search with more hyperparameters in some range and previous result:

The model we saved into a pkl file.

As we can see below, only `n_estimators` decreased from 140 up to 120, the rest of parameters left the same as first time from Fig.8 above:

```

XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
               colsample_bynode=1, colsample_bytree=1, gamma=1, gpu_id=-1,
               importance_type='gain', interaction_constraints=None,
               learning_rate=0.1, max_delta_step=0, max_depth=2,
               min_child_weight=1, missing=nan, monotone_constraints=None,
               n_estimators=120, n_jobs=0, num_parallel_tree=1,
               objective='binary:logistic', random_state=34000, reg_alpha=0,
               reg_lambda=1, scale_pos_weight=0.1767388699628784, seed=34000,
               subsample=1, tree_method=None, validate_parameters=False,
               verbosity=None)

```

And in third run we tuned `reg_lambda`, `reg_alpha`, :

```

XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
               colsample_bynode=1, colsample_bytree=0.7, gamma=1, gpu_id=-1,
               importance_type='gain', interaction_constraints=None,
               learning_rate=0.1, max_delta_step=0, max_depth=2,
               min_child_weight=1, missing=nan, monotone_constraints=None,
               n_estimators=120, n_jobs=0, num_parallel_tree=1,
               objective='binary:logistic', random_state=34000, reg_alpha=0.05,
               reg_lambda=0.8, scale_pos_weight=0.1767388699628784, seed=34000,
               subsample=1, tree_method=None, validate_parameters=False,
               verbosity=None)

```

## 5.Conclusions

- We ran `predict_proba` on Validation set
- Detected TP and FP
- Assigned TP as 1 FP as 0
- Ran our second model on this chosen set
- Got results and improved them by tuning the model with GridSearch

With all our efforts we saw not high accuracy 82% after model tuning

## 6. References & Links

1. Pierre Singer, Dr. Itai Bendavid, Dr. Liran Statlender, Dr. Leonid Shvartser, Shmuel Teppler, Roy Azullay, Rotem Sapir: Prediction of Invasive Mechanical Ventilation following Noninvasive Mechanical Ventilation Failure in patients suffering from COVID-19 respiratory failure using Machine Learning Models, CRITICAL CARE JOURNAL (12.07.2020)
2. Prof. Pierre Singer, Dr. Itai Bendavid, Dr. Liran Statlender, Dr. Leonid Shvartser, Shmuel Teppler. Roy Azullay, Rotem Sapir: Prediction of Noninvasive Mechanical Ventilation Failure and need for Invasive Mechanical Ventilation, Based on MIMIC-III Data using Machine Learning Models, TSG IT Advanced Systems Ltd. (22.4.2020)
3. Gitlab repository [https://gitlab.com/amproyGit/mimic/-/tree/Galina\\_Dev](https://gitlab.com/amproyGit/mimic/-/tree/Galina_Dev)