



TSG IT Systems

report

Improvement of quality of Prediction of Mechanical

Ventilation by

**prove that False Positives produced by the Vent “sliding
window” algorithm are not real False Positives**

Galina Blokh 29/09/2020

1. Introduction

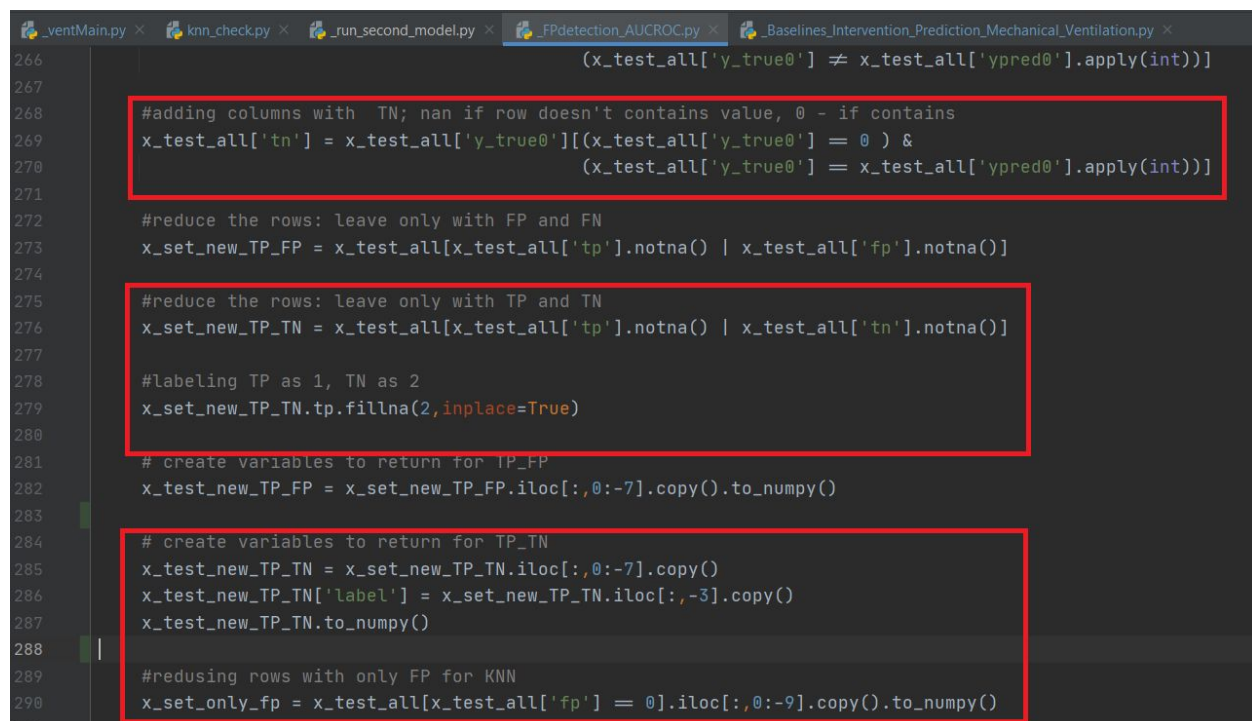
The idea of this short exercise is to prove that False Positives produced by the Vent “sliding window” algorithm are not real False Positives. They have the features closer to True Positives than to True Negatives. It means that they are taken from the intervals close to the True Positive ones or from the patients that finally did not deteriorate to Mechanical Ventilation, but were rather close to it in their clinical parameters.

The scheme is as follows:

After running first XGBoost on Train set and prediction on Test set

2.1. To put all the True Positives feature intervals in Class 1 and all the True Negatives feature intervals in Class 2.

For this task in `C:\Projects\mimic\MIMIC_Extract_tsg\src\Vent\utils_FPdetection_AUCROC.py` were added several lines of code to label our predicted classes from TP and TN into class1 and class2 accordingly:



```
266 (x_test_all['y_true0'] != x_test_all['ypred0']).apply(int))
267
268 #adding columns with TN; nan if row doesn't contains value, 0 - if contains
269 x_test_all['tn'] = x_test_all['y_true0'][(x_test_all['y_true0'] == 0) &
270 (x_test_all['y_true0'] == x_test_all['ypred0']).apply(int))]
271
272 #reduce the rows: leave only with FP and FN
273 x_set_new_TP_FP = x_test_all[x_test_all['tp'].notna() | x_test_all['fp'].notna()]
274
275 #reduce the rows: leave only with TP and TN
276 x_set_new_TP_TN = x_test_all[x_test_all['tp'].notna() | x_test_all['tn'].notna()]
277
278 #labeling TP as 1, TN as 2
279 x_set_new_TP_TN.tp.fillna(2,inplace=True)
280
281 # create variables to return for TP_FP
282 x_test_new_TP_FP = x_set_new_TP_FP.iloc[:,0:-7].copy().to_numpy()
283
284 # create variables to return for TP_TN
285 x_test_new_TP_TN = x_set_new_TP_TN.iloc[:,0:-7].copy()
286 x_test_new_TP_TN['label'] = x_set_new_TP_TN.iloc[:, -3].copy()
287 x_test_new_TP_TN.to_numpy()
288
289 #reducing rows with only FP for KNN
290 x_set_only_fp = x_test_all[x_test_all['fp'] == 0].iloc[:,0:-9].copy().to_numpy()
```

Pic.1. Creating working set for KNN detection

We created a set with TP, FP, FN and TN variables as features.

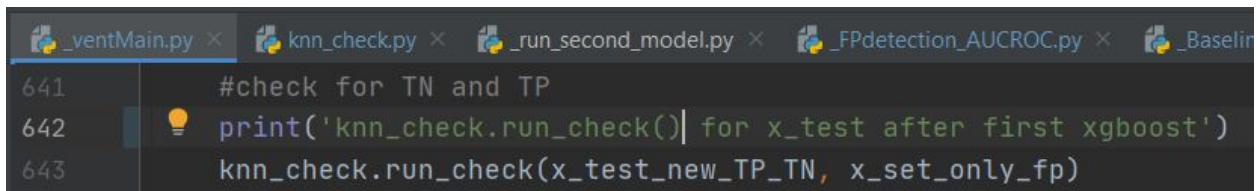
Then we marked TP as '1' and TN as '2' and reduced rows with all features which are in TP and TN marked areas.

In last lines of the *Pic.1* , we see code for creating a separate ndarray for the next step in 2.2

2.2 To pass the False Positives feature intervals one by one. To check its Nearest Neighbor or K-Nearest Neighbors. To affiliate every interval to Class 1 or Class 2 according to these checks.

In C:\Projects\mimic\MIMIC_Extract_tsg\src\Vent_OnSet_Sheba_FalseP_detection_ventMain.py in lines 646-643 we run a new created function from new created python document

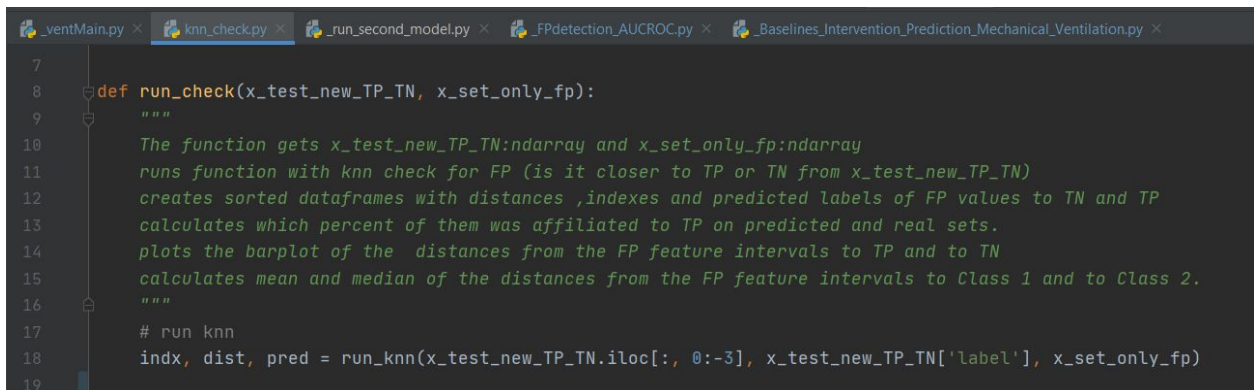
C:\Projects\mimic\MIMIC_Extract_tsg\src\Vent_OnSet_Sheba_FalseP_detection\knn_check.py



```
641 #check for TN and TP
642 print('knn_check.run_check()' for x_test after first xgboost')
643 knn_check.run_check(x_test_new_TP_TN, x_set_only_fp)
```

Pic.2 Run check for TP and TN in main

Here we send a big set with all features from the TP and TN area and a set with only FP lines from the test set. The big set also contain predicted + true labels, which we will drop later from big set and will use them to make calculations in step 2.3



```
7
8 def run_check(x_test_new_TP_TN, x_set_only_fp):
9     """
10    The function gets x_test_new_TP_TN:ndarray and x_set_only_fp:ndarray
11    runs function with knn check for FP (is it closer to TP or TN from x_test_new_TP_TN)
12    creates sorted dataframes with distances ,indexes and predicted labels of FP values to TN and TP
13    calculates which percent of them was affiliated to TP on predicted and real sets.
14    plots the barplot of the distances from the FP feature intervals to TP and to TN
15    calculates mean and median of the distances from the FP feature intervals to Class 1 and to Class 2.
16    """
17    # run knn
18    indx, dist, pred = run_knn(x_test_new_TP_TN.iloc[:, 0:-3], x_test_new_TP_TN['label'], x_set_only_fp)
19
```

Pic.3 New function for prove that False Positives produced by the Vent “sliding window” algorithm are not real False Positives

In the last line 18 on Pic.3 we call a function with KNN algorithm.

First was an idea to use an unsupervised version of this algorithm because it was giving less lines of code, but with going deeper in scikit-learn documentation we understood that supervised method gives back more information and more metrics to make a better full analysis.

On Pic.4 we see that in KNN algorithm we are looking for only one neighbor, due to the task itself “To pass the False Positives feature intervals one by one. To check its Nearest Neighbor or K-Nearest Neighbors”

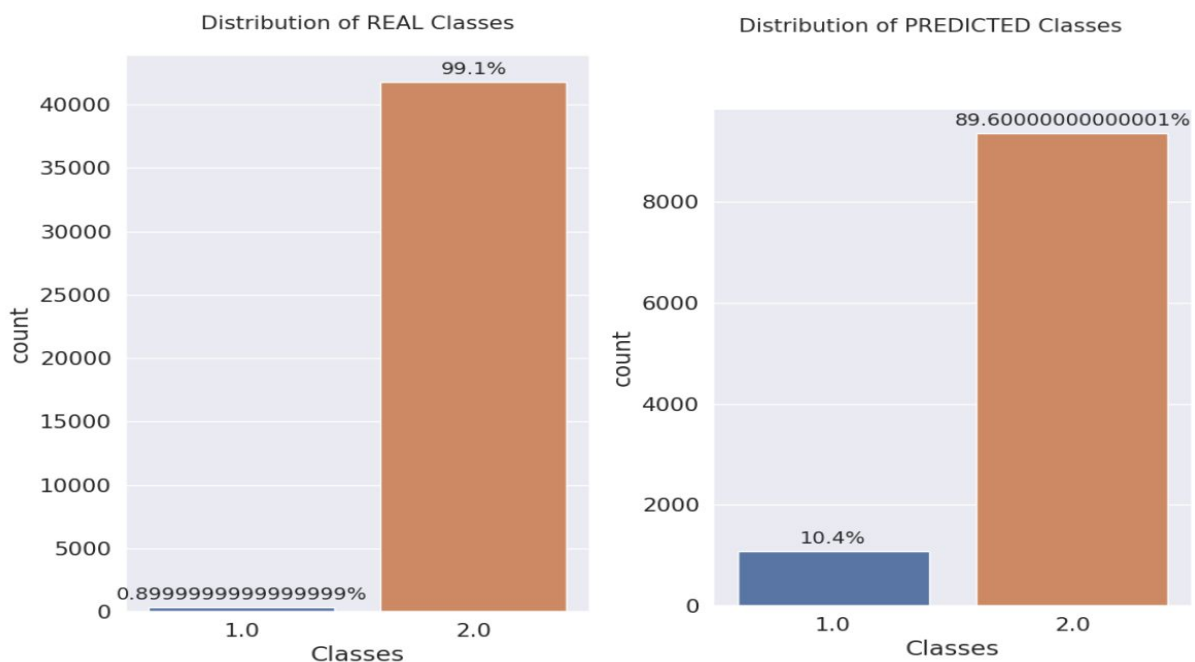
```

87 def run_knn(X, y, fp):
88     """
89     The function get x_set: ndarray, y_labels, fp_set: ndarray,
90     run KNeighborsClassifier with n_neighbors=1 and algorithm = 'auto'
91     return euclidean distances: ndarray, indices: ndarray, predicted labels
92     """
93     nbrs = KNeighborsClassifier(n_neighbors=1).fit(np.nan_to_num(X), y)
94     predict = nbrs.predict(np.nan_to_num(fp))
95
96     distances, indxs = nbrs.kneighbors(np.nan_to_num(fp), return_distance=True)
97     return indxs, distances, predict

```

Pic.4 run_knn() function with one neighbor

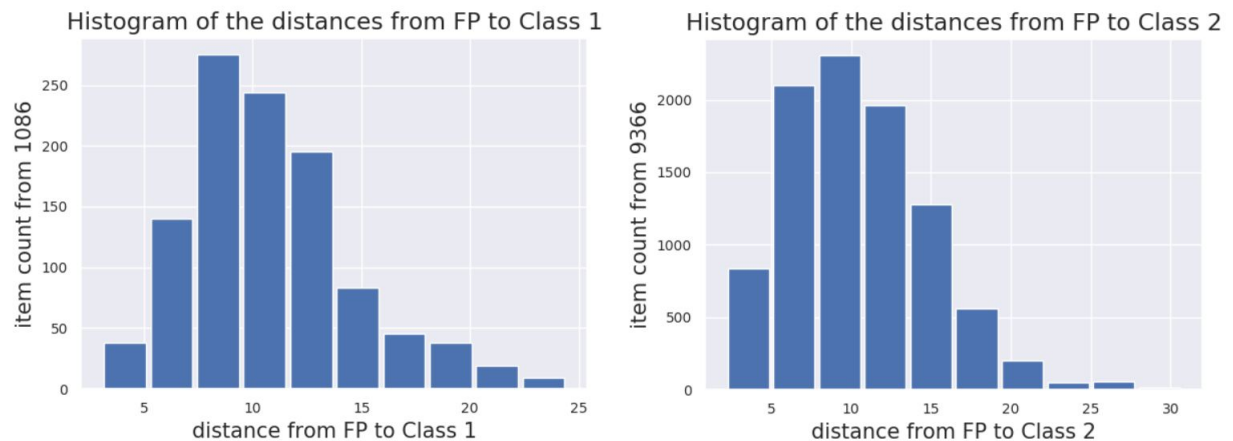
2.3 To calculate which percent of them was affiliated to Class 1.



Pic.5 Comparing histograms of true labels (after first XGBoost - left) and labels after KNN run (right)

On the Pic.5 we see that our KNN method detected a bigger % of TP than it was made by simple XGBoost run.

2.4 To build the histograms of the distance from the FP feature intervals to Class 1 and to Class 2.



Pic.6 Comparing histogram of the distance from the FP feature intervals to Class 1 and to Class 2

From Pic.6 we see that the distribution is right skewed for both classes, but skewed less for the Class1.

There are more items from Class2 with diapazone of its distances in [2-27].

For Class1 this diapazone is less - [2-25].

2.5 To calculate the mean and median of the shortest distance from the FP feature intervals to Class 1 and to Class 2.

Mean distance from FP to Class 1: 10.8

Mean distance from FP to Class 2: 10.48

Median distance from FP to Class 1: 10.17

Median distance from FP to Class 2: 10.0

3. Conclusions

From 2.5, 2.4 and 2.3 it is clear that:

- according to mean and median distance values, our FP values are closer to TN than to TP
- TN and TP points have almost the same distance diapazone