```
!unzip data_input41.zip
```

```
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2

  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
```

```
  inflating: data_input41/01_healthy/image_tumor_patient15012020_633nm_obj2
```

```
!pip install catboost
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
Collecting catboost
  Downloading catboost-1.1.1-cp38-none-manylinux1_x86_64.whl (76.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.6/76.6 MB 12.4 MB/s eta 0:00
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.8/dis
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.8/dist
Requirement already satisfied: plotly in /usr/local/lib/python3.8/dist-packag
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-package
Requirement already satisfied: graphviz in /usr/local/lib/python3.8/dist-pack
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-pa
Requirement already satisfied: six in /usr/local/lib/python3.8/dist-packages
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/pytho
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /u
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.8/di
Installing collected packages: catboost
Successfully installed catboost-1.1.1
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os
import keras
from keras.models import Sequential
from keras.layers import Flatten, Dense, Lambda
from keras.layers import Convolution2D
from keras.layers.pooling import MaxPooling2D
from keras.layers import Cropping2D
from keras.layers.core import Dense, Dropout, Activation
from keras.layers import Dense, Conv1D, Flatten, MaxPooling1D
from keras import regularizers
from keras.layers import BatchNormalization
from keras.utils import np_utils
from keras.utils import plot_model
import graphviz, pydot, pydotplus
from sklearn.model_selection import train_test_split
import tensorflow as tf


import os
os.environ["CUDA_VISIBLE_DEVICES"]="0,1"


print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

```
Num GPUs Available:  1
```

```
root_folder = "data_input41/01_healthy"
healthy = []
```

```
for root, dirs, files in os.walk(root_folder):
    for filename in files:
        healthy.append(filename)


measurements = []
classifications = []
count_healhy = 0
for sample in healthy:
    df = pd.read_csv(os.path.join(root_folder,sample), sep='\t',skiprows=[0],
        header=None, names=[ 'Wave', 'Intensity'])
    measurements.append(df[['Intensity']].to_numpy())
    classifications.append([1,0])
    count_healhy += 1

print(f'Здоровые ткани: {count_healhy}')
```

```
Здоровые ткани: 432
```

```
sick_path = 'data_input41/image_tumor_patient20022019_633nm.txt'
count_sick = 0
df = pd.read_csv(sick_path, sep='\t',skiprows=[0],
    header=None, names=['X', 'Y', 'Wave', 'Intensity'])
for i in range(456):
    measurements.append(df[['Intensity']][i*len(df['Wave'].unique()):(i+1)*len(df[
    classifications.append([0,1])
    count_sick += 1

print(f'Больные ткани: {count_sick}')
print(f'Всего образцов: {count_healhy + count_sick}')
```

```
Больные ткани: 456
Всего образцов: 888
```

```
X = np.asarray(measurements)
y = np.asarray(classifications)


X.shape
```

```
(888, 1015, 1)
```

```
X = X.reshape(888, 1015)
X.shape
```

```
(888, 1015)
```

```
y.shape
```

```
(888, 2)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.24, random_s
```

```
print(f'Total amount of train measurements: {X_train.shape}')
print(f'Total amount of train labels: {y_train.shape}')
print(f'Total amount of test measurements: {X_test.shape}')
print(f'Total amount of test labels: {y_test.shape}')
y_train_labels = np.argmax(y_train, axis=1)
y_test_labels = np.argmax(y_test, axis=1)
```

```
Total amount of train measurements: (674, 1015)
Total amount of train labels: (674, 2)
Total amount of test measurements: (214, 1015)
Total amount of test labels: (214, 2)
```

Precision - доля объектов, названных классификатором положительными и при этом действительно являющимися положительными

Recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

F1 - среднее гармоническое precision и recall

```
from keras import backend as K

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))


# Model 1
model = Sequential()
model.add(Conv1D(128, 4, activation='relu', input_shape=(1015,1),kernel_regularize
model.add(Conv1D(128, 4, activation='relu', bias_regularizer=regularizers.l2(1e-4)
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling1D())
model.add(Dropout(0.25))
model.add(Conv1D(256, 2, activation='relu', kernel_regularizer=regularizers.l1_l2(
model.add(Conv1D(256, 2, activation='relu', bias_regularizer=regularizers.l2(1e-4)
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling1D())
model.add(Dropout(0.25))
model.add(Flatten())
```

```
model.add(Dense(256, activation = 'relu', use_bias=False))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dense(128, activation = 'relu', use_bias=False))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dense(64, activation = 'relu', use_bias=False))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(Dense(2, activation = 'softmax'))
model.summary()
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accurac
model_hist = model.fit(X_train, y_train, batch_size=64, epochs=40, verbose=1)
```

```
Non-trainable params: 1,664
_________________________________________________________________
Epoch 1/40
11/11 [==============================] - 11s 61ms/step - loss: 0.8272 - acc
Epoch 2/40
11/11 [==============================] - 0s 44ms/step - loss: 0.7395 - accu
Epoch 3/40
11/11 [==============================] - 0s 44ms/step - loss: 0.7752 - accu
Epoch 4/40
11/11 [==============================] - 0s 44ms/step - loss: 0.6717 - accu
Epoch 5/40
11/11 [==============================] - 0s 45ms/step - loss: 0.6549 - accu
Epoch 6/40
11/11 [==============================] - 0s 45ms/step - loss: 0.6135 - accu
Epoch 7/40
11/11 [==============================] - 0s 45ms/step - loss: 0.6302 - accu
Epoch 8/40
11/11 [==============================] - 0s 44ms/step - loss: 0.5846 - accu
Epoch 9/40
11/11 [==============================] - 0s 45ms/step - loss: 0.6044 - accu
Epoch 10/40
11/11 [==============================] - 0s 45ms/step - loss: 0.6063 - accu
Epoch 11/40
11/11 [==============================] - 0s 44ms/step - loss: 0.5462 - accu
Epoch 12/40
11/11 [==============================] - 0s 44ms/step - loss: 0.5672 - accu
Epoch 13/40
11/11 [==============================] - 0s 45ms/step - loss: 0.5644 - accu
Epoch 14/40
11/11 [==============================] - 0s 44ms/step - loss: 0.5499 - accu
Epoch 15/40
11/11 [==============================] - 0s 45ms/step - loss: 0.5523 - accu
Epoch 16/40
11/11 [==============================] - 0s 44ms/step - loss: 0.5274 - accu
Epoch 17/40
11/11 [==============================] - 0s 45ms/step - loss: 0.5518 - accu
Epoch 18/40
11/11 [==============================] - 0s 44ms/step - loss: 0.5556 - accu
Epoch 19/40
11/11 [==============================] - 0s 45ms/step - loss: 0.5970 - accu
Epoch 20/40
11/11 [==============================] - 0s 45ms/step - loss: 0.5547 - accu
Epoch 21/40
11/11 [==============================] - 0s 44ms/step - loss: 0.5543 - accu
Epoch 22/40
```

```
11/11 [==============================] - 0s 44ms/step - loss: 0.5520 - accu
Epoch 23/40
11/11 [==============================] - 0s 45ms/step - loss: 0.5601 - accu
Epoch 24/40
11/11 [==============================] - 0s 44ms/step - loss: 0.5331 - accu
Epoch 25/40
11/11 [==============================] - 0s 45ms/step - loss: 0.5579 - accu
Epoch 26/40
11/11 [==============================] - 0s 44ms/step - loss: 0.5494 - accu
Epoch 27/40
11/11 [==============================] - 0s 45ms/step - loss: 0.5306 - accu
Epoch 28/40
11/11 [==============================] - 0s 45ms/step - loss: 0.5161 - accu
Epoch 29/40
```

```
acc = model.evaluate(X_test, y_test)
print("Loss:", acc[0], " Accuracy:", acc[1], " F1 :", acc[2])
pred = model.predict(X_test)
print(np.round(pred,2)[0], np.round(pred,2)[1])
#model.save("model1.h5")
```

```
7/7 [==============================] - 0s 13ms/step - loss: 0.7416 - accuracy
Loss: 0.7415792942047119  Accuracy: 0.5467289686203003  F1 : 0.53855514526367
7/7 [==============================] - 0s 10ms/step
[0.5 0.5] [0.54 0.46]
```

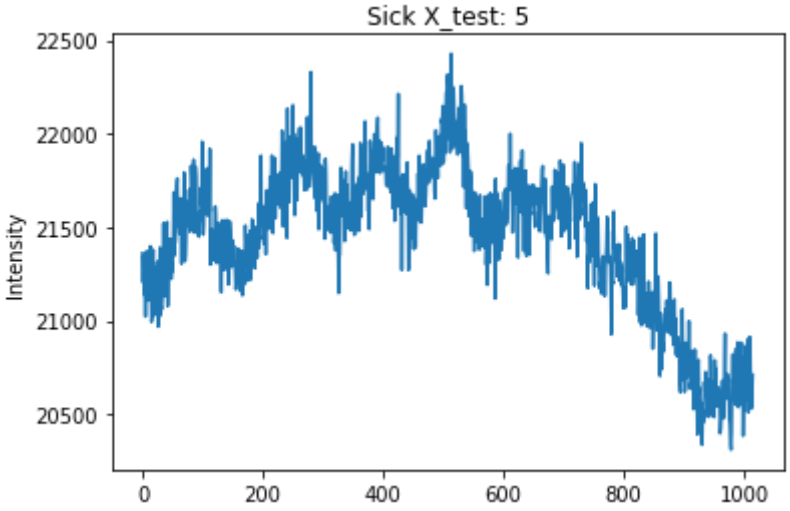
```
np.round(pred,2)
```

```
[0.46, 0.54],
[0.57, 0.43],
[1.  , 0.  ],
[0.82, 0.18],
[0.51, 0.49],
[0.81, 0.19],
[0.62, 0.38],
[0.57, 0.43],
[0.54, 0.46],
[0.59, 0.41],
[0.65, 0.35],
[0.56, 0.44],
[0.52, 0.48],
[0.58, 0.42],
[0.57, 0.43],
[0.68, 0.32],
[0.46, 0.54],
[0.93, 0.07],
[0.68, 0.32],
[0.58, 0.42],
[0.84, 0.16],
[0.54, 0.46],
[0.54, 0.46],
[0.53, 0.47],
[0.47, 0.53],
[0.56, 0.44],
[0.5 , 0.5 ],
[0.72, 0.28],
[1.  , 0.  ],

[0.57, 0.43],
```

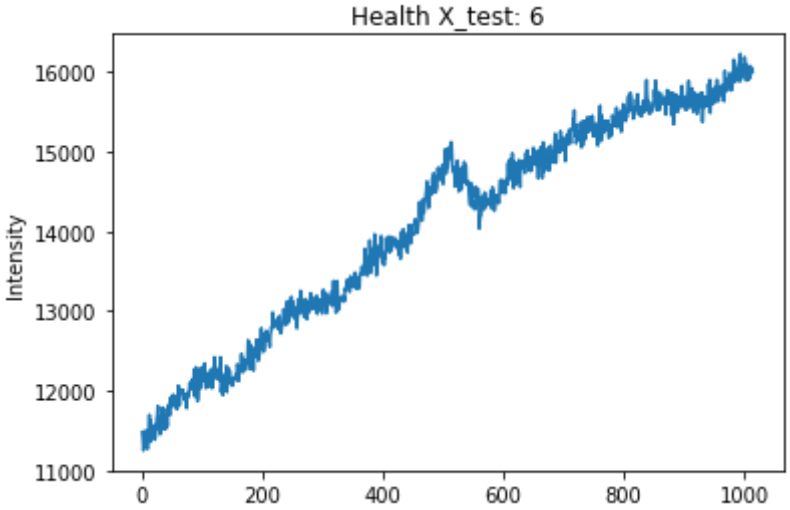
```
       [0.59, 0.41],
       [0.55, 0.45],
       [0.48, 0.52],
       [0.48, 0.52],
       [0.5 , 0.5 ],
       [0.58, 0.42],
       [0.78, 0.22],
       [0.82, 0.18],
       [0.46, 0.54],
       [0.46, 0.54],
       [0.73, 0.27],
       [0.49, 0.51],
       [0.55, 0.45],
       [0.8 , 0.2 ],
       [0.59, 0.41],
       [0.69, 0.31],
       [0.55, 0.45],
       [0.79, 0.21],
       [0.59, 0.41],
       [0.5 , 0.5 ],
       [0.56, 0.44],
       [0.62, 0.38],
       [0.88, 0.12],
       [0.79, 0.21],
       [0.83, 0.17],
       [0.6 , 0.4 ],
       [0.55, 0.45],
       [0.51, 0.49],
       [0.98, 0.02],
       [0.49, 0.51]
```

```
pred = model.predict(X_test)
for i in range(5,20):
    sick_true = "Sick" if (y_test[i] == [0, 1]).all() else "Health"
    sick_pred = "Sick" if pred[i].argmax() else "Health"
    print(f'TrueLabel: {sick_true}, PredLabel {sick_pred}; PredProb: {np.round(pre
    plt.plot(X_test[i])
    plt.title(f"{sick_true} X_test: {i}")
    plt.ylabel("Intensity")

    plt.show()
```

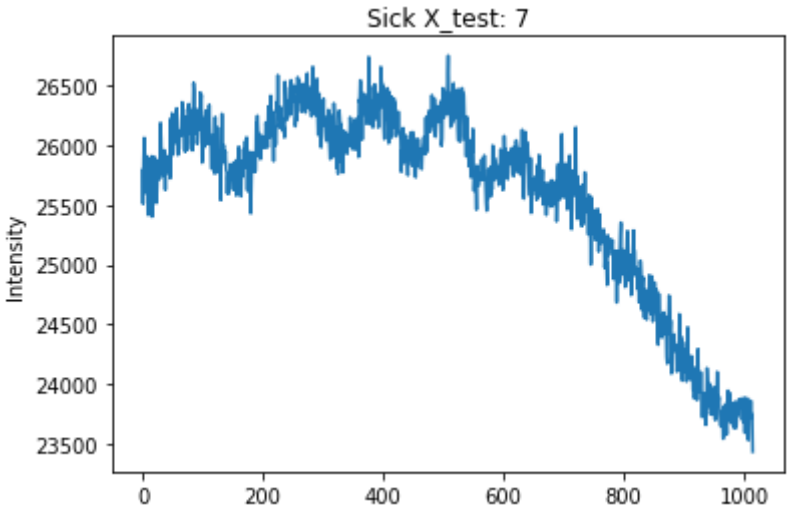
```
7/7 [==============================] - 0s 12ms/step
TrueLabel: Sick, PredLabel Health; PredProb: [0.52 0.48], TrueProb [0 1]
```



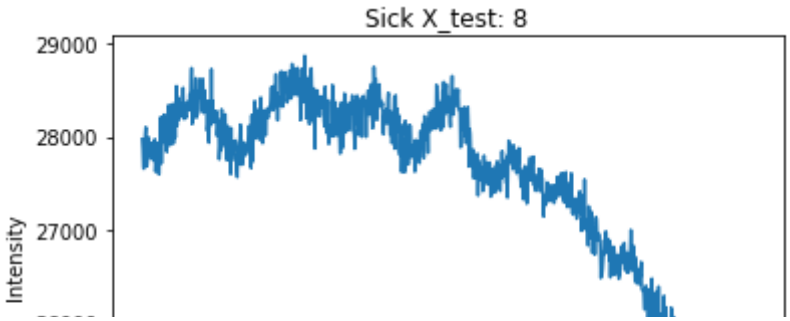

```
TrueLabel: Health, PredLabel Health; PredProb: [0.55 0.45], TrueProb [1 0]
```

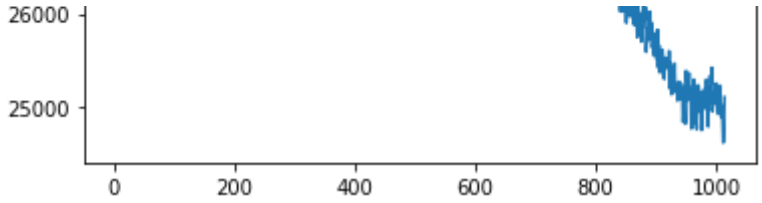



```
TrueLabel: Sick, PredLabel Sick; PredProb: [0.47 0.53], TrueProb [0 1]
```

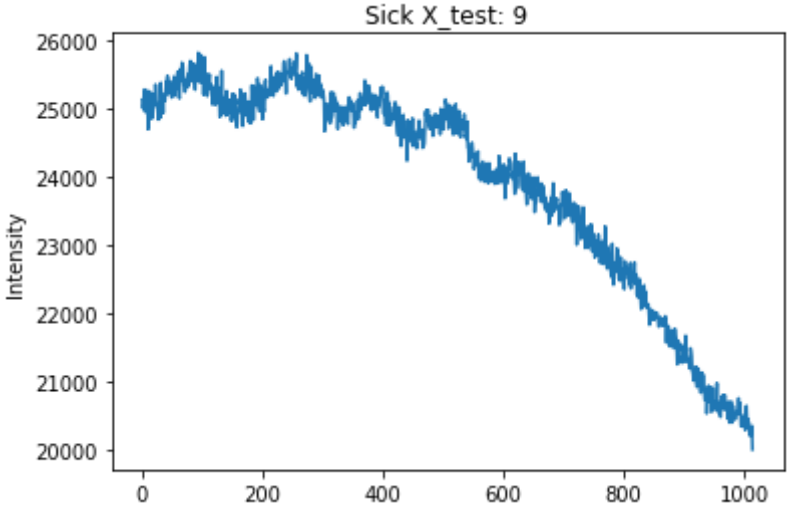



```
TrueLabel: Sick, PredLabel Health; PredProb: [0.55 0.45], TrueProb [0 1]
```

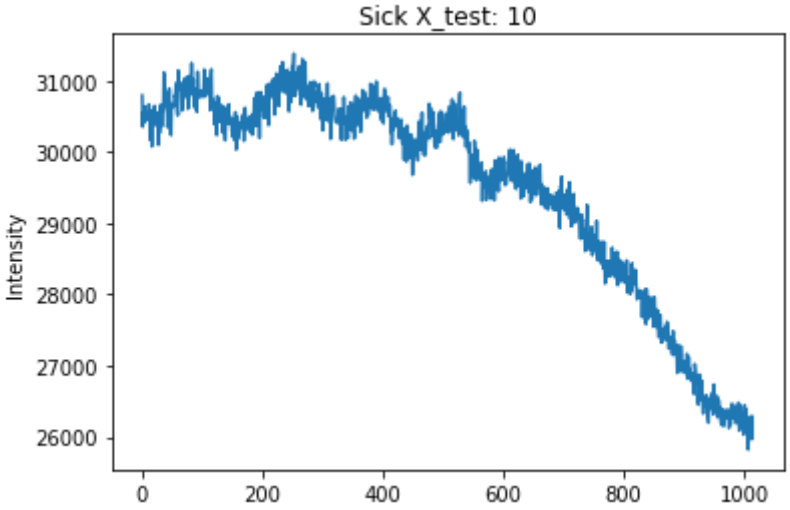




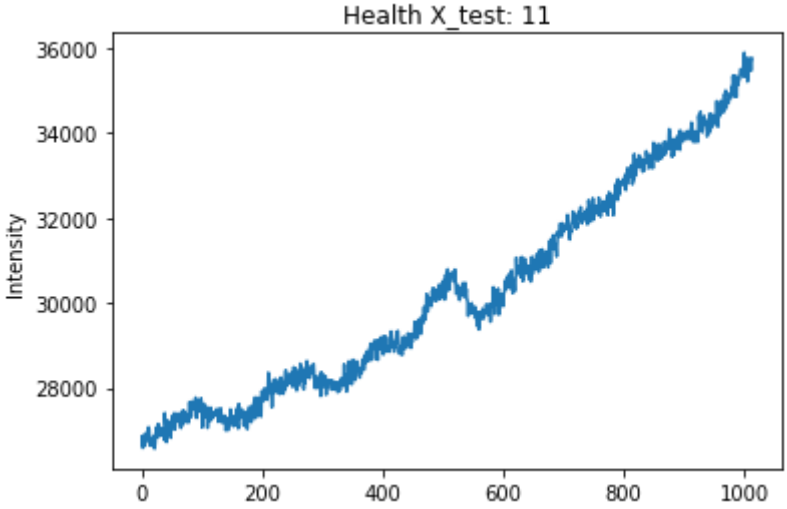
TrueLabel: Sick, PredLabel Sick; PredProb: [0.47 0.53], TrueProb [0 1]



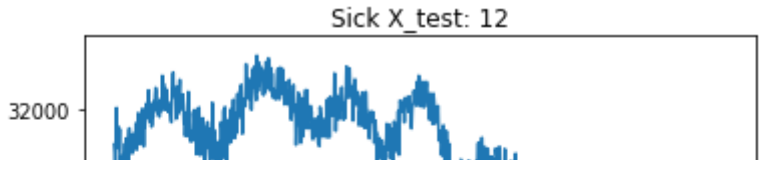
TrueLabel: Sick, PredLabel Health; PredProb: [0.6 0.4], TrueProb [0 1]

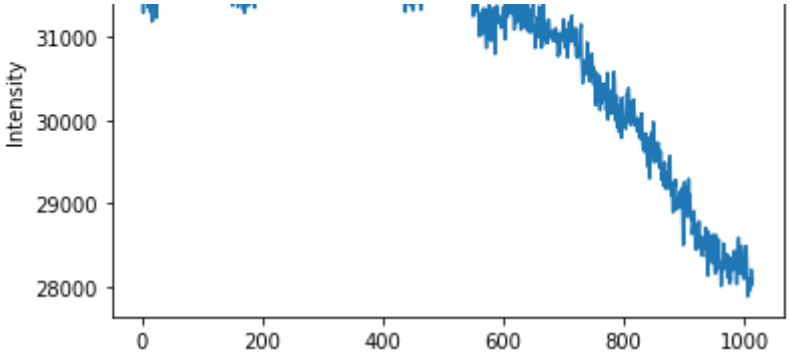


TrueLabel: Health, PredLabel Health; PredProb: [0.58 0.42], TrueProb [1 0]

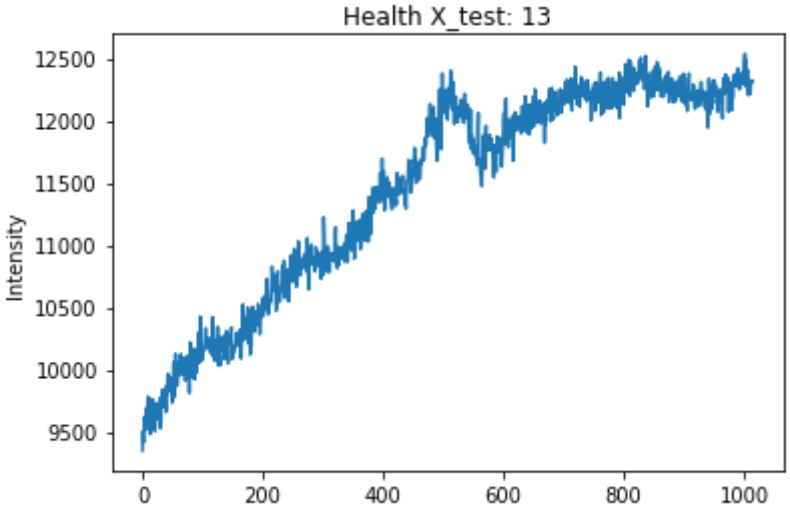


TrueLabel: Sick, PredLabel Health; PredProb: [0.81 0.19], TrueProb [0 1]

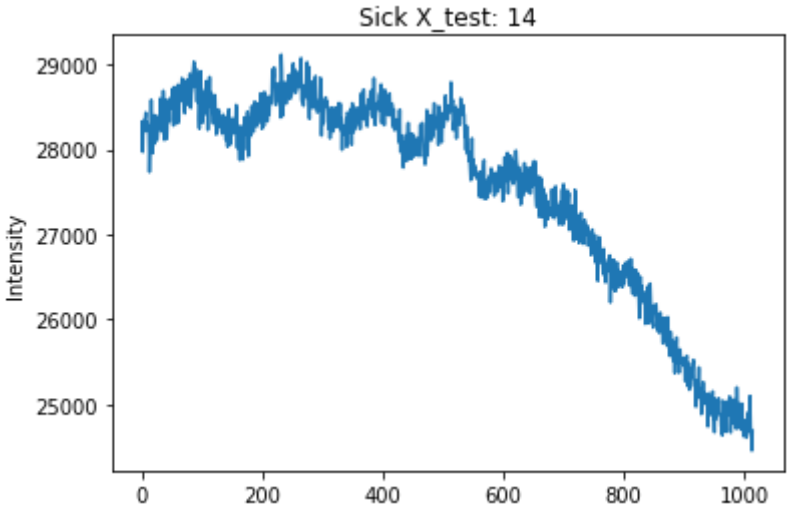




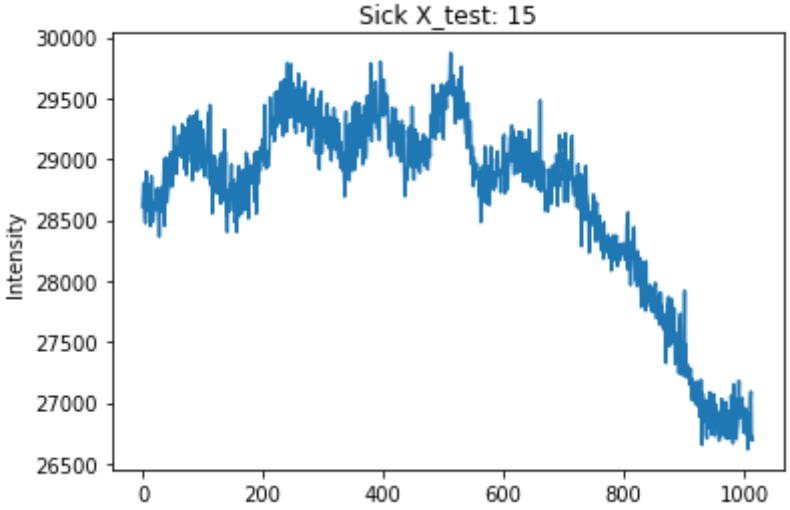
TrueLabel: Health, PredLabel Health; PredProb: [0.56 0.44], TrueProb [1 0]



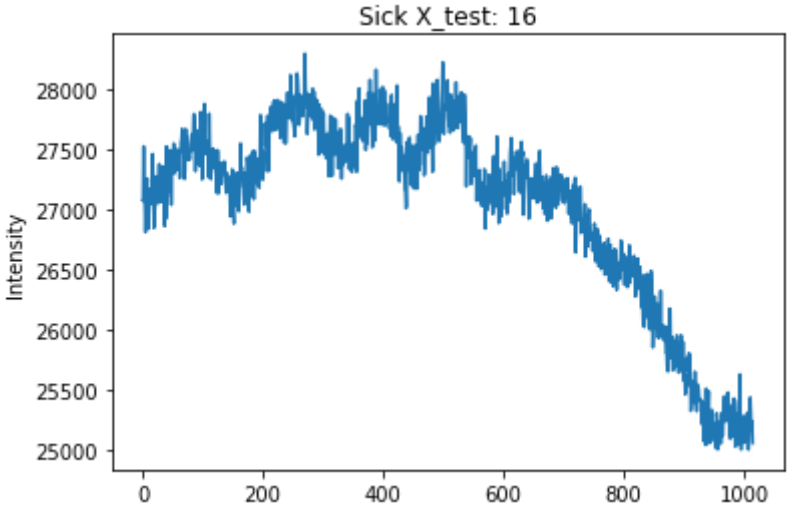
TrueLabel: Sick, PredLabel Health; PredProb: [0.55 0.45], TrueProb [0 1]



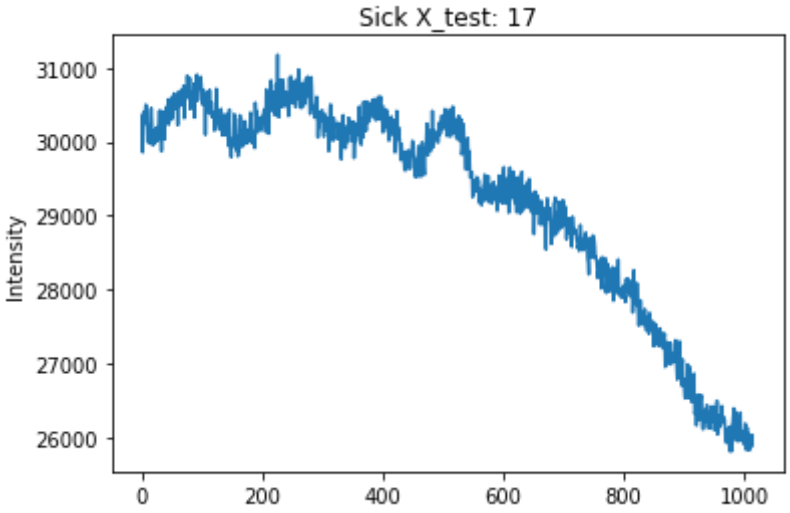
TrueLabel: Sick, PredLabel Health; PredProb: [0.59 0.41], TrueProb [0 1]



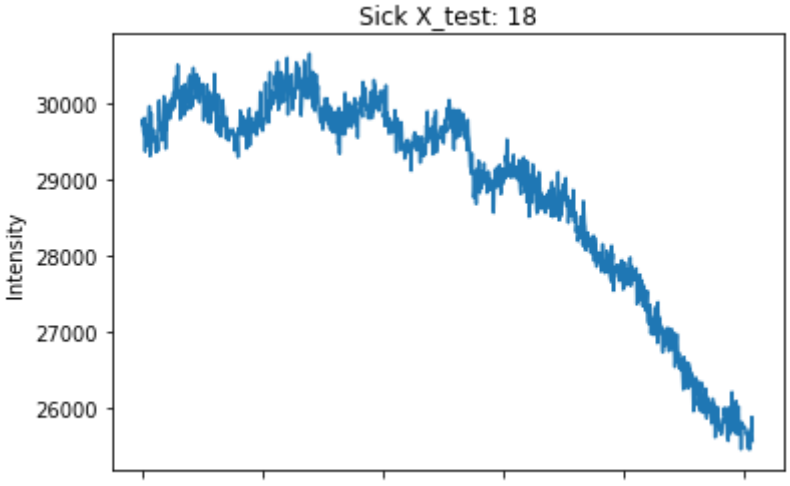
TrueLabel: Sick, PredLabel Health; PredProb: [0.53 0.47], TrueProb [0 1]



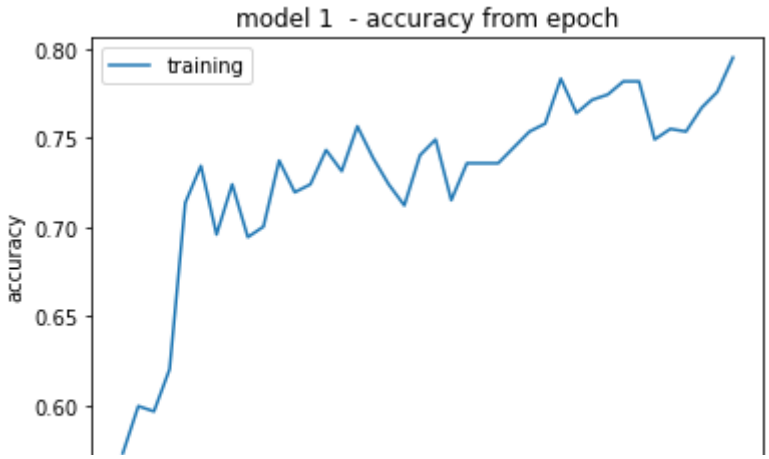

```
TrueLabel: Sick, PredLabel Health; PredProb: [0.59 0.41], TrueProb [0 1]
```




```
TrueLabel: Sick, PredLabel Health; PredProb: [0.59 0.41], TrueProb [0 1]
```




```
loss, accuracy, f1  = model.evaluate(X_test, y_test, verbose=False)

plt.plot(model_hist.history['accuracy'])
plt.title('model 1  - accuracy from epoch')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['training', 'validation'], loc='best')
plt.show()
```



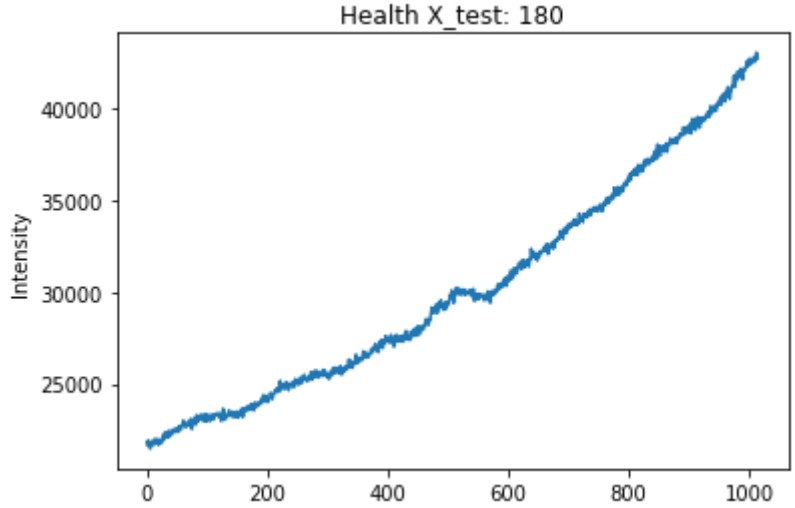
CatBoost - градиентный бустинг от Яндекса, подбирает гиперпараметры модели самостоятельно, т.е. можно получить хороший результат без предварительной настройки

```
from catboost import CatBoostClassifier
from sklearn.metrics import f1_score
model_catboost = CatBoostClassifier(verbose=False)
model_catboost.fit(X_train, y_train_labels)
print(f'CatBoost F1 Score {f1_score(y_test_labels, model_catboost.predict(X_test))

from sklearn.metrics import accuracy_score
print(f'CatBoost Accuracy {accuracy_score(y_test_labels, model_catboost.predict(X_
```

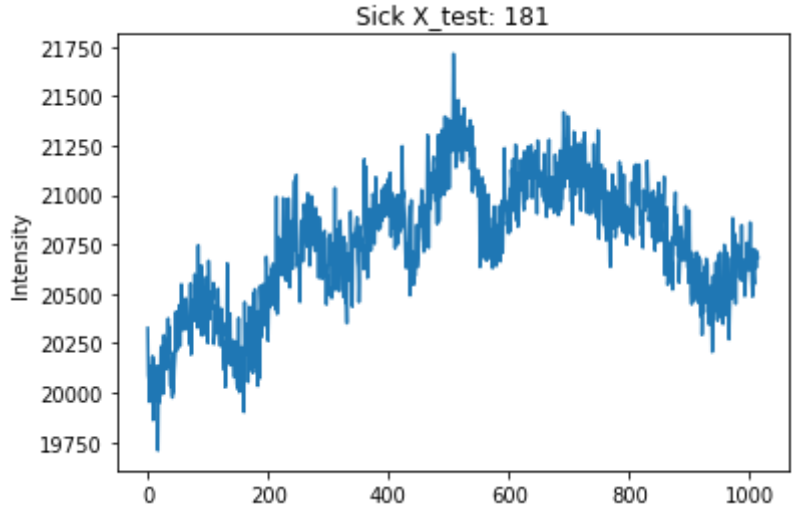
```
CatBoost F1 Score 0.920353982300885
CatBoost Accuracy 0.9158878504672897
```

```
pred = model_catboost.predict_proba(X_test)
for i in range(180,189):
    sick_true = "Sick" if (y_test[i] == [0, 1]).all() else "Health"
    sick_pred = "Sick" if pred[i].argmax() else "Health"
    print(f'TrueLabel: {sick_true}, PredLabel {sick_pred}; PredProb: {np.round(pre
    plt.plot(X_test[i])
    plt.title(f"{sick_true} X_test: {i}")
    plt.ylabel("Intensity")

    plt.show()
```

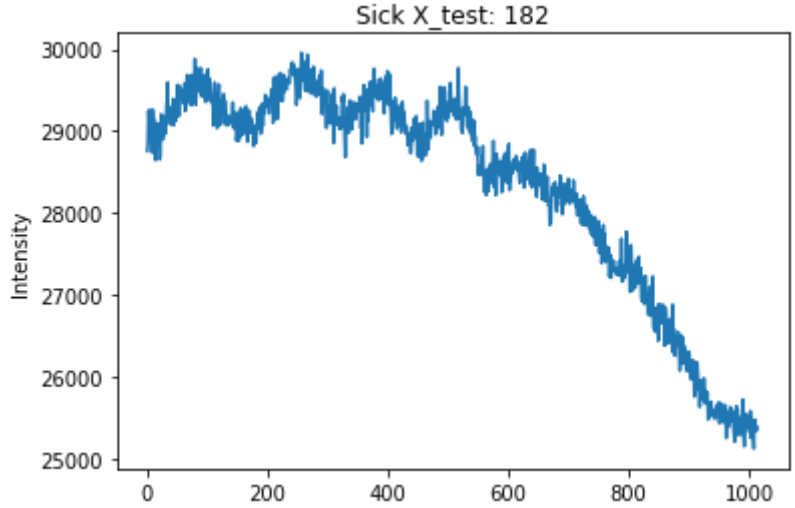
```
TrueLabel: Health, PredLabel Health; PredProb: [0.99 0.01], TrueProb [1 0]
```



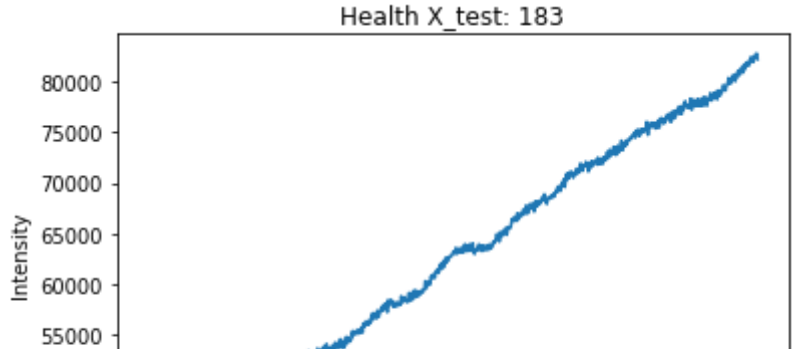

```
TrueLabel: Sick, PredLabel Sick; PredProb: [0.14 0.86], TrueProb [0 1]
```

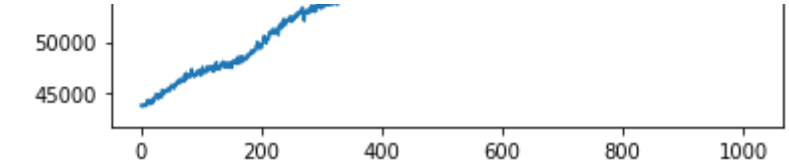



```
TrueLabel: Sick, PredLabel Sick; PredProb: [0.01 0.99], TrueProb [0 1]
```

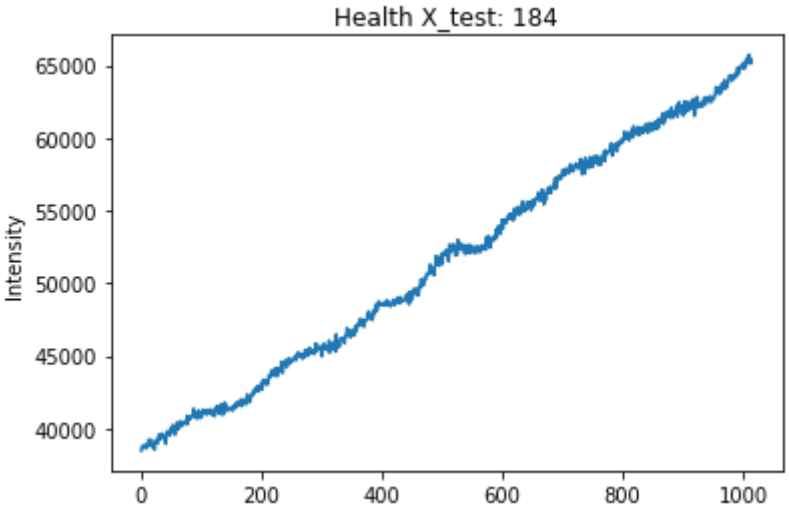



```
TrueLabel: Health, PredLabel Health; PredProb: [1. 0.], TrueProb [1 0]
```



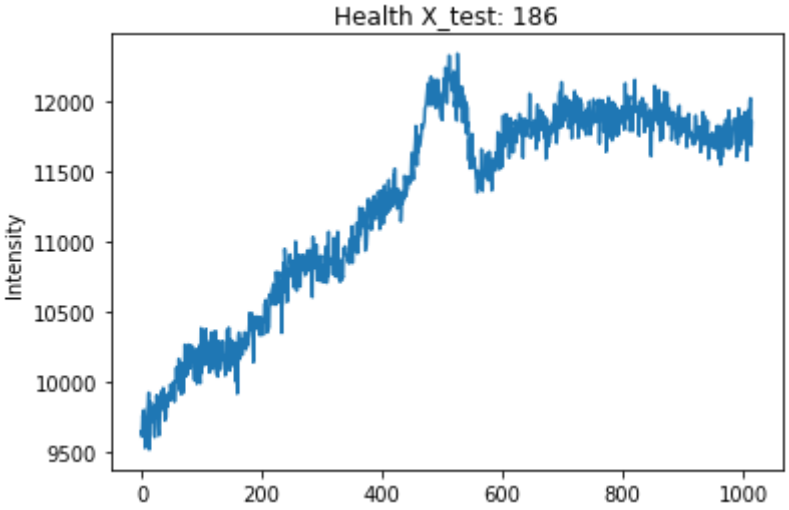


TrueLabel: Health, PredLabel Health; PredProb: [1. 0.], TrueProb [1 0]

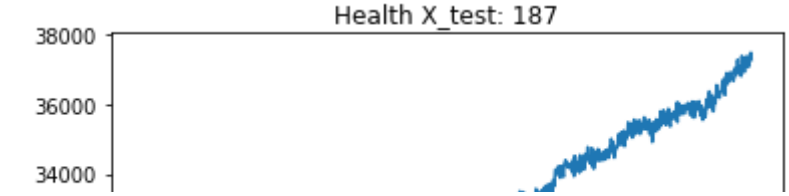


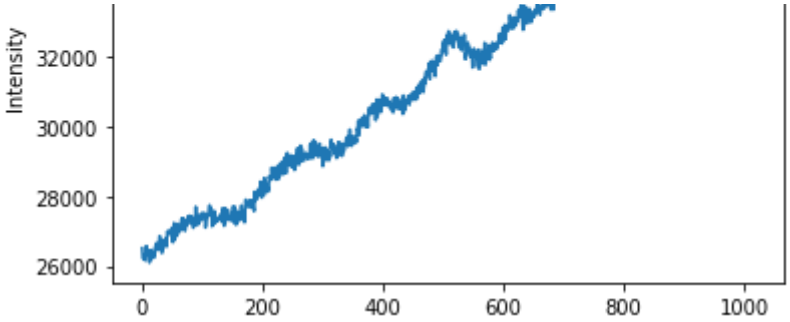
TrueLabel: Health, PredLabel Health; PredProb: [0.99 0.01], TrueProb [1 0]

TrueLabel: Health, PredLabel Health; PredProb: [0.91 0.09], TrueProb [1 0]

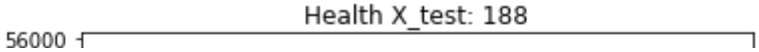


TrueLabel: Health, PredLabel Health; PredProb: [0.99 0.01], TrueProb [1 0]

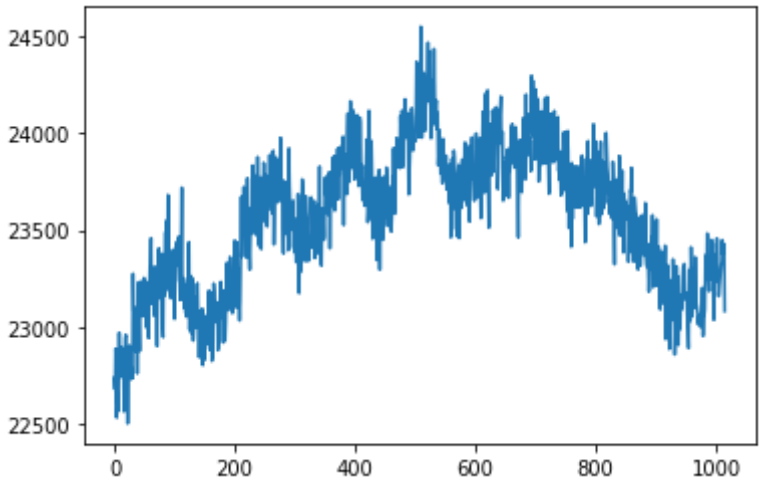




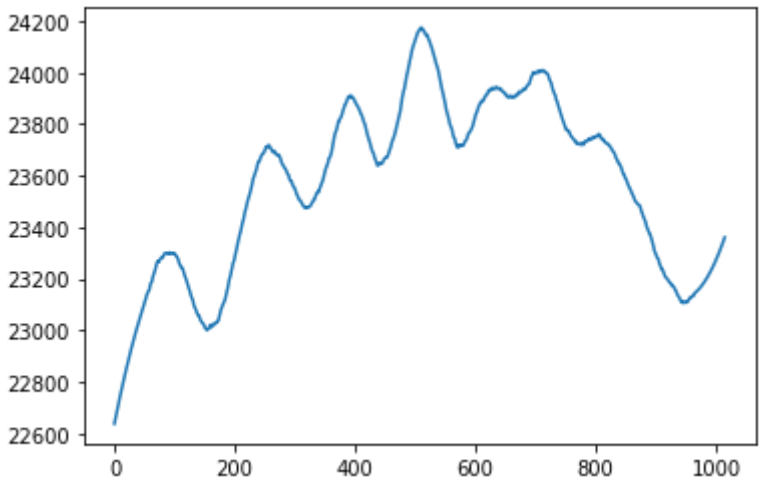

```
TrueLabel: Health, PredLabel Health; PredProb: [1. 0.], TrueProb [1 0]
```




```
plt.plot(X_train[20])
plt.show()
```




```
from scipy.signal import savgol_filter
w = savgol_filter(X_train[20], 101, 2)
plt.plot(w)  # high frequency noise removed
plt.show()
```



Savitzky-Golay filter (Фильтр Савицки-Голея) - фильтр для сглаживания данных. Параметр window_length - нечетное число, чем больше чем сглаженне данные

Аналоги которые можно попробовать: Kalman filter, IIR filter, LOWESS

- Kalman filter

- IIR filter
- LOWESS (Locally Weighted Scatterplot Smoothing)
- rolling mean
- RBF (radial basis function) interpolation

```
from scipy.signal import savgol_filter
for i in range(len(X_train)):
    X_train[i] = savgol_filter(X_train[i], 101, 2)
```

```
from keras import backend as K

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

```
# Model 2
model2 = Sequential()
model2.add(Conv1D(128, 4, activation='relu', input_shape=(1015,1),kernel_regulariz
model2.add(Conv1D(128, 4, activation='relu', bias_regularizer=regularizers.l2(1e-4
model2.add(BatchNormalization())
model2.add(Activation('relu'))
model2.add(MaxPooling1D())
model2.add(Dropout(0.25))
model2.add(Conv1D(256, 2, activation='relu', kernel_regularizer=regularizers.l1_l2
model2.add(Conv1D(256, 2, activation='relu', bias_regularizer=regularizers.l2(1e-4
model2.add(BatchNormalization())
model2.add(Activation('relu'))
model2.add(MaxPooling1D())
model2.add(Dropout(0.25))
model2.add(Flatten())
model2.add(Dense(256, activation = 'relu', use_bias=False))
model2.add(BatchNormalization())
model2.add(Activation('relu'))
model2.add(Dense(128, activation = 'relu', use_bias=False))
model2.add(BatchNormalization())
model2.add(Activation('relu'))
model2.add(Dense(64, activation = 'relu', use_bias=False))
model2.add(BatchNormalization())
model2.add(Activation('relu'))
```

```
model2.add(Dropout(0.25))
model2.add(Dense(2, activation = 'softmax'))
model2.summary()
model2.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accura
model2_hist = model2.fit(X_train, y_train, batch_size=128, epochs=100, verbose=1)
```

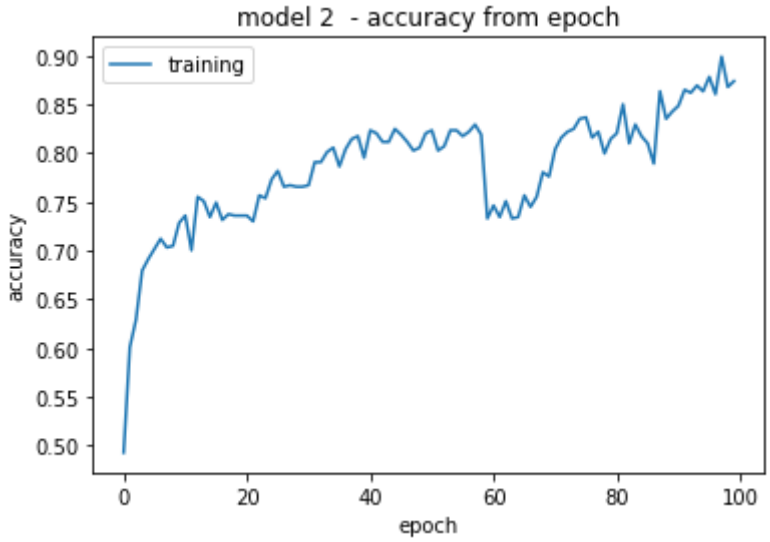
```
6/6 [==============================] - 1s 82ms/step - loss: 0.4709 - accura
Epoch 71/100
6/6 [==============================] - 0s 82ms/step - loss: 0.4661 - accura
Epoch 72/100
6/6 [==============================] - 0s 82ms/step - loss: 0.4526 - accura
Epoch 73/100
6/6 [==============================] - 0s 82ms/step - loss: 0.4168 - accura
Epoch 74/100
6/6 [==============================] - 0s 81ms/step - loss: 0.3972 - accura
Epoch 75/100
6/6 [==============================] - 1s 82ms/step - loss: 0.4005 - accura
Epoch 76/100
6/6 [==============================] - 1s 81ms/step - loss: 0.3815 - accura
Epoch 77/100
6/6 [==============================] - 0s 81ms/step - loss: 0.3859 - accura
Epoch 78/100
6/6 [==============================] - 0s 82ms/step - loss: 0.3735 - accura
Epoch 79/100
6/6 [==============================] - 0s 82ms/step - loss: 0.4042 - accura
Epoch 80/100
6/6 [==============================] - 1s 83ms/step - loss: 0.3815 - accura
Epoch 81/100
6/6 [==============================] - 0s 81ms/step - loss: 0.3664 - accura
Epoch 82/100
6/6 [==============================] - 0s 82ms/step - loss: 0.3312 - accura
Epoch 83/100
6/6 [==============================] - 0s 81ms/step - loss: 0.3939 - accura
Epoch 84/100
6/6 [==============================] - 0s 82ms/step - loss: 0.3398 - accura
Epoch 85/100
6/6 [==============================] - 0s 81ms/step - loss: 0.3601 - accura
Epoch 86/100
6/6 [==============================] - 0s 82ms/step - loss: 0.3448 - accura
Epoch 87/100
6/6 [==============================] - 0s 81ms/step - loss: 0.3961 - accura
Epoch 88/100
6/6 [==============================] - 0s 81ms/step - loss: 0.3160 - accura
Epoch 89/100
6/6 [==============================] - 1s 82ms/step - loss: 0.3145 - accura
Epoch 90/100
6/6 [==============================] - 0s 80ms/step - loss: 0.3275 - accura
Epoch 91/100
6/6 [==============================] - 0s 81ms/step - loss: 0.3088 - accura
Epoch 92/100
6/6 [==============================] - 0s 81ms/step - loss: 0.2856 - accura
Epoch 93/100
6/6 [==============================] - 0s 81ms/step - loss: 0.2797 - accura
Epoch 94/100
6/6 [==============================] - 0s 81ms/step - loss: 0.2974 - accura
Epoch 95/100
6/6 [==============================] - 0s 81ms/step - loss: 0.2799 - accura
Epoch 96/100
6/6 [==============================] - 0s 81ms/step - loss: 0.2757 - accura
Epoch 97/100
6/6 [==============================] - 0s 81ms/step - loss: 0.2797 - accura
```

```
Epoch 98/100
6/6 [==============================] - 0s 81ms/step - loss: 0.2552 - accura
Epoch 99/100
```

```
acc = model2.evaluate(X_test, y_test)
print("Loss:", acc[0], " Accuracy:", acc[1], " F1 :", acc[2])
pred = model2.predict(X_test)
print(np.round(pred,2)[0], np.round(pred,2)[1])
print(X_test[22])
#model2.save("model2.h5")
```

```
7/7 [==============================] - 0s 13ms/step - loss: 0.6731 - accuracy
Loss: 0.6731101870536804  Accuracy: 0.663551390171051  F1 : 0.672483742237091
7/7 [==============================] - 0s 10ms/step
[0.97 0.03] [0.45 0.55]
[21382.988281 21374.837891 21435.46875  ... 39662.578125 39570.792969
 39699.359375]
```

```
loss, accuracy, f1  = model2.evaluate(X_test, y_test, verbose=False)

plt.plot(model2_hist.history['accuracy'])
plt.title('model 2  - accuracy from epoch')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['training', 'validation'], loc='best')
plt.show()
```




```
from catboost import CatBoostClassifier
from sklearn.metrics import f1_score, accuracy_score
model_catboost = CatBoostClassifier(verbose=False)
model_catboost.fit(X_train, y_train_labels)
print(f'CatBoost F1 Score {f1_score(y_test_labels, model_catboost.predict(X_test))
print(f'CatBoost Accuracy {accuracy_score(y_test_labels, model_catboost.predict(X_
```

```
CatBoost F1 Score 0.9339207048458149
CatBoost Accuracy 0.9299065420560748
```

```
!pip install tsmoothie
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
Collecting tsmoothie
  Downloading tsmoothie-1.0.4-py3-none-any.whl (21 kB)
Collecting simdkalman
  Downloading simdkalman-1.0.2-py2.py3-none-any.whl (11 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-package
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-package
Installing collected packages: simdkalman, tsmoothie
Successfully installed simdkalman-1.0.2 tsmoothie-1.0.4
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_s
print(f'Total amount of train measurements: {X_train.shape}')
print(f'Total amount of train labels: {y_train.shape}')
print(f'Total amount of test measurements: {X_test.shape}')
print(f'Total amount of test labels: {y_test.shape}')
y_train_labels = np.argmax(y_train, axis=1)
y_test_labels = np.argmax(y_test, axis=1)
```

```
Total amount of train measurements: (666, 1015)
Total amount of train labels: (666, 2)
Total amount of test measurements: (222, 1015)
Total amount of test labels: (222, 2)
```

```
import numpy as np
import matplotlib.pyplot as plt
from tsmoothie.smoother import *

# operate smoothing
smoother = ConvolutionSmoother(window_len=30, window_type='ones')
smoother.smooth(X_train)

# generate intervals
low, up = smoother.get_intervals('sigma_interval', n_sigma=3)

plt.figure(figsize=(11,6))
plt.plot(smoother.data[21], color='orange')
plt.plot(smoother.smooth_data[21], linewidth=3, color='blue')
plt.fill_between(range(len(smoother.data[21])), low[21], up[21], alpha=0.2)
```