





TASK 2

MEDICAL EVENTS PREDICTION

You are in the group  Sklearn consisting of  bosliu (bosliu@student.ethz.ch (mailto://bosliu@student.ethz.ch)),  hanwan (hanwan@student.ethz.ch (mailto://hanwan@student.ethz.ch)) and  junzhe (junzhe@student.ethz.ch (mailto://junzhe@student.ethz.ch)).

1. READ THE TASK DESCRIPTION

2. SUBMIT SOLUTIONS

3. HAND IN FINAL SOLUTION

1. TASK DESCRIPTION

INTRODUCTION

Applying ML approaches to real-world problems requires some extra steps in handling specific data artifacts. Some common challenges you may face are missing values, an imbalance of the labels, or heavy-tailed distributions in the data. These can be addressed through a specific choice of pre-processing and/or model components.

In this task, as an illustration of a real-world problem, you are asked to predict the evolution of hospital patients' states and needs during their stay in the Intensive Care Unit (ICU). For each patient, you are provided with the monitoring information collected during the first 12h of their stay. From the data, you first need to extract meaningful features describing this 12h stay. Then, for each sub-task, you should select an appropriate model to train on your pre-processed data. You will face the typical challenges of working with real medical data: missing features and imbalanced classification, predicting rarely-occurring events

In the following two sections, we give more details on the dataset and sub-tasks you have to solve.

DATASET DESCRIPTION

[Download handout \(/static/task2_k49am2lqi.zip\)](/static/task2_k49am2lqi.zip)

The handout you download contains the following files:

- **train_features.csv** training set features
- **train_labels.csv** training set labels
- **test_features.csv** test set features
- **sample.zip** a sample submission file in the correct format, compressed
- **score_submission.py** code to illustrate how the server calculates the score for a submission

The training (train_features.csv) and test set (test_features.csv) are both arranged in the same way. Each row contains the information for a patient at a given time identified by a unique patient ID and timestamp in columns *pid* and *Time*. The rest of the columns are medical information, either vital signs such as the *Heart rate*, or lab tests such as *Calcium* level in the patient blood. Finally, you are provided with the *Age* of the patient which is the same during the entire stay.

All medical measurements are not available at each timestep, meaning the data contains **a lot of missing values**, indicated with 'nan' in our case. To simplify the problem, the data is already **re-sampled hourly**. This means that we aggregate measurements by one-hour period, thus there are only **12 rows for a given patient** in the corresponding .csv file.

The last .csv file, namely train_labels.csv, contains the ground truth labels of the training set for each subtask. Each row is identified by a unique pid corresponding to a patient from the training set.

SUB-TASKS DESCRIPTION

We now detail the three types of sub-tasks you are asked to solve. In all sub-tasks, we want to predict an aspect of the evolution of a patient's state from the first 12 hours of his stay in the ICU.

SUB-TASK 1: ORDERING OF MEDICAL TEST

Here we are interested in anticipating the future needs of the patient. You have to predict whether a certain medical test is ordered by a clinician in the remaining stay. This sub-task is a **binary classification** : 0 means that there will be no further tests of this kind ordered whereas 1 means that at least one is ordered in the remaining stay.

The corresponding columns containing the binary ground truth in train_labels.csv are: *LABEL_BaseExcess*, *LABEL_Fibrinogen*, *LABEL_AST*, *LABEL_Alkalinephos*, *LABEL_Bilirubin_total*, *LABEL_Lactate*, *LABEL_TroponinI*, *LABEL_SaO2*, *LABEL_Bilirubin_direct*, *LABEL_EtCO2*.

Because there is an imbalance between labels in these sub-tasks we evaluate the performance of a model with the Area Under the Receiver Operating Characteristic Curve (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html#sklearn.metrics.roc_auc_score), which is a threshold-based metric. To achieve good performance, it is important to **produce (probabilistic) real-valued predictions in the interval [0, 1]**.

SUB-TASK 2: SEPSIS PREDICTION

In this sub-task, we are interested in anticipating future life-threatening events. You have to predict whether a patient is likely to have a sepsis (<https://en.wikipedia.org/wiki/Sepsis>) event in the remaining stay. This task is also a **binary classification** : 0 means that no sepsis will occur, 1 otherwise.

The corresponding column containing the binary ground-truth in train_labels.csv is *LABEL_Sepsis*.

This task is also imbalanced, thus we'll also evaluate performance using Area Under the Receiver Operating Characteristic Curve (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html#sklearn.metrics.roc_auc_score).

SUB-TASK 3: KEYS VITALS SIGNS PREDICTION

In this type of sub-task, we are interested in predicting a more general evolution of the patient state. To this effect, here we aim at predicting the mean value of a vital sign in the remaining stay. This is a **regression** task.

The corresponding columns containing the real-valued ground truth in train_labels.csv are: *LABEL_RRate*, *LABEL_ABPM*, *LABEL_SpO2*, *LABEL_Heartrate*.

To evaluate the performance of a given model on this sub-task we use R^2 Score (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html).

EVALUATION

For each type of sub-task, when multiple instances exist (1 and 3) we take the average performance over them. For the regression sub-task 3, we measure the R^2 score (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html), threshold it below at 0 and

normalize it to the range $[0.5, 1]$. Your final score is the average of these 3 scores. For your convenience, in the handout we provide you with the `get_score` function in `score_submission.py`, which shows how to calculate this score in Python.

Note: for subtasks 1 and 2, you will need to produce predictions in the interval $[0, 1]$. How can you achieve this with an SVM? In the lecture, you have seen that the SVM prediction for binary classification is $\text{sign}(\sum_{i=1}^n \alpha_i y_i k(x, x_i))$. In order to produce real-valued predictions in the interval $[0, 1]$ with SVM, you can replace the `sign` function by the sigmoid function $\sigma(x) = 1/(1 + e^{-x})$.

The submission format should be identical to the format of `sample.csv` compressed in `sample.zip`, which is provided in the handout.

Both `train_features.csv` and `test_features.csv` contain missing values ('nan' entries). An important part of this project is how to deal with such missing data (known as 'data imputation' in the ML literature) and feature engineering (what features can you extract from measurements taken in consecutive hours, etc.)

Important:

- While `train_features.csv` and `test_features.csv` contain multiple entries for a single patient (identified by the `pid`), your submission should contain a single line associated with each `pid`, since you are predicting vital signs / events / medical test orders for a patient for the period between the 13th recorded hour and the end of their hospital stay. Please see `sample.zip` for reference.
- To keep the server load small, there is an 1 MB total upload limit for the submission. For this, please submit your predictions in a **zip** archive. To reduce the size further, we recommend to limit the decimal places you write for floats. An example of how to write a pandas dataframe directly to a zip, with floats up to 3 decimal digits:

```
import pandas as pd

# suppose df is a pandas dataframe containing the result
df.to_csv('prediction.zip', index=False, float_format='%.3f', compression='zip')
```

GRADING

We provide you with **one test set** for which you have to compute predictions. We have partitioned this test set into two parts (of the same size) and use it to compute a *public* and a *private* score for each submission. You only receive feedback about your performance on the public part in the form of the public score, while the private leaderboard remains secret. The purpose of this division is to prevent overfitting to the public score. Your model should generalize well to the private part of the test set. When handing in the task, you need to select which of your submissions will get graded and provide a short description of your approach. This has to be done **individually by each member** of the team. We will then compare your selected submission to our baseline. This project task is graded with either **pass (6.0)**, **partial pass (4.0)** or **fail (2.0)**. To fully pass the project (grade: 6.0), you need to perform better than the baseline in both private and public score. If you only outperform the baseline in either the private or the public score, you will get a partial pass (grade: 4.0). In addition, for the pass/fail decision, we consider the code and the description of your solution that you submitted. The following **non-binding** guidance provides you with an idea on what is expected to pass the project: If you hand in a properly-written description, your source code is runnable and reproduces your predictions, and your submission performs better than the baseline, you can expect to have passed the assignment.

⚠ Make sure that you properly hand in the task, otherwise you may obtain zero points for this task.

FREQUENTLY ASKED QUESTIONS

🕒 WHICH PROGRAMMING LANGUAGE AM I SUPPOSED TO USE? WHAT TOOLS AM I ALLOWED TO USE?

You are free to choose any programming language and use any software library. However, **we strongly encourage you to use Python**. You can use publicly available code, but you should specify the source as a comment in your code.

🕒 AM I ALLOWED TO USE MODELS THAT WERE NOT TAUGHT IN THE CLASS?

Yes. Nevertheless, the baselines were designed to be solvable based on the material taught in the class up to the second week of each task.

🕒 IN WHAT FORMAT SHOULD I SUBMIT THE CODE?

You can submit it as a single file (main.py, etc.; you can compress multiple files into a .zip) having max. size of 1 MB. If you submit a zip, please make sure to name your main file as *main.py* (possibly with other extension corresponding to your chosen programming language).

🕒 WILL YOU CHECK / RUN MY CODE?

We will check your code and compare it with other submissions. We also reserve the right to run your code. Please make sure that your code is runnable and your predictions are reproducible (fix the random seeds, etc.). Provide a readme if necessary (e.g., for installing additional libraries).

🕒 SHOULD I INCLUDE THE DATA IN THE SUBMISSION?

No. You can assume the data will be available under the path that you specify in your code.

🕒 CAN YOU HELP ME SOLVE THE TASK? CAN YOU GIVE ME A HINT?

As the tasks are a graded part of the class, **we cannot help you solve them**. However, feel free to ask general questions about the course material during or after the exercise sessions.

🕒 CAN YOU GIVE ME A DEADLINE EXTENSION?

⚠️ We do not grant any deadline extensions!

🕒 CAN I POST ON MOODLE AS SOON AS HAVE A QUESTION?

This is highly discouraged. Remember that collaboration with other teams is prohibited. Instead,

- Read the details of the task thoroughly.
- Review the frequently asked questions.
- If there is another team that solved the task, spend more time thinking.
- Discuss it with your team-mates.

🕒 WHEN WILL I RECEIVE THE PRIVATE SCORES? AND THE PROJECT GRADES?

We will publish the private scores, and corresponding grades before the exam the latest.