TECHNISCHE UNIVERSITÄT DRESDEN

Universitätsklinikum Carl Gustav Carus
DIE DRESDNER.

Zentrum für Medizinische Informatik

Documentation

Concerning software:	PyETL & DecubiTection
Submitted by :	Alex Jenke, Markus Fritzsche
Submitted on:	February 06 2020

Contents

1 Description **1**
1.1 ETL process . . . 1
1.2 Decubitus Risk Prediction . . . 5
1.3 Front-End . . . 5

2 Risk Analysis **5**

3 Quality Management **8**

4 Software Life-cycle Processes **8**
4.1 Requirements . . . 8
4.2 Tests . . . 9
4.3 Change Management . . . 9

5 Usability **10**
5.1 Front-End . . . 10
5.2 Back-End . . . 12

6 Dependencies **13**
6.1 ETL-Process . . . 13
6.2 Decubitus Risk Prediction . . . 13
6.3 Front-End . . . 13

1 Description

This documentation describes the software: *PyETL & DecubiTection*, a software transforming CSV data into an OMOP common data model and using machine learning to identify potential decubitus risks, regarding hospital patients.

The software is divided into three main systems: *ETL process*, *Machine Learning / Decubitus risk prediction* and *Front-end*. Each system is able to run on its own and only depends on the results of the preceding subsystem stored in a database.

Figure 1: Overview over the subsystems and their relations

There is the possibility to schedule the ETL-Process and the decubitus detection algorithm in a certain interval through a cron job. For this purpose, a Python script is provided which is able to create a cron job with the required parameters such as database host, password, etc.

1.1 ETL process

The purpose of the ETL-Process is to map medical data, provided by a set of CSV files formatted according to the specifications of §21 Paragraphs 4 and 5 KHEntgG, into the OMOP common data model (CDM). This step is required, since the later described machine learning system learns and predicts, using data, gathered from an OMOP CDM.

The ETL-Process requires certain preconditions. The CSV files are required to only contain valid and new data. Otherwise, the ETL-Process might fail or inconsistent data will be inserted into the OMOP database.

The following explanation describes the ETL-Process for a single patient. In the first step, data from the *FALL.csv* file gets extracted and will be inserted in the OMOP database, according to table 1. Since the provided data does not contain information about the patient's race and ethnicity, default values will be inserted. In case the patient's location is not already stored in the database, the ETL-Process will supplement this information, as depicted in table 2. If in two or more visits, the information differs, only the most up-to-date information will be inserted. For every row in the FALL.csv file, a new visit will be inserted into VISIT_OCCURRENCE as shown in table 3. If the care site, the patient is currently located in, is not stored in the CARE_SITE table, a new record is added, as illustrated in table 4.

In the next step, all values - concerning the patient - from the *LABOR.csv* file will be inserted, as depicted in table 5. As the file only contains measurements, rows containing a different domain ID than "Measurement" will be skipped. If that happens, a warning will be printed onto STDERR. Similarly, the ETL-Process maps the data from *MEASUREMENT.csv* to the database, as illustrated in table 6.

The data from the *ICD.csv* file is mapped as follows. If the domain ID of a row is an "Observation", a new observation will be inserted into OBSERVATION. If the value of "sekundaer_kode"

is empty, only one data record will be added as depicted in table 7, in which case "ICD.icd_kode" is mapped to observation_source_value. If "sekundaer_kode" is not empty, a second record will be added, where "ICD.sekundaer_kode" is mapped to observation_source_value.

If the domain ID is a "Condition", instead of an "Observation", a new condition will be inserted in the database, as depicted by table 11. Similar to the previously described case, if a "sekundaer_kode" is available, another record will be added. This is illustrated in table 11. Additionally, if "lokalisation" is not empty, a new record will be inserted in the database, as shown in table 8. The same applies if "diagnosesicherheit" is not empty, in which case the data is mapped as depicted in table 10. Both records will be connected to the previously generated records through a fact relationship in both directions, as illustrated in table 9. If the domain id is neither "Condition" nor "Observation", the row will be skipped and a warning will be printed.

From *OPS.csv*, the ETL-Process maps all procedures - concerning the patient - to the database, as illustrated in table 12. Equally as described earlier, if a row contains a domain id, different from "Procedure", the row will be skipped and a warning will be printed.

source column	**destination column**
FALL.patienten_nummer	PERSON.person_id
FALL.last_record	PERSON.gender_concept_id
FALL.geburtsjahr	PERSON.year_of_birth
FALL.geburtsmonat	PERSON.month_of_birth
"Unknown"	PERSON.race_concept_id
"Not Hispanic"	PERSON.ethnicity_concept_id
FALL.gender_source_value	PERSON.geschlecht
LOCATION.location_id	PERSON.location

Table 1: Insert data from FALL.csv into table PERSON.

source column	**destination column**
FALL.city	LOCATION.city
FALL.zip	LOCATION.zip

Table 2: Insert a patient's location from FALL.csv to LOCATION.

source column	**destination column**
FAB.FAB	CARE_SITE.care_site_name

Table 3: Insert a patient's visits to VISIT_OCCURRENCE.

source column	destination column
FALL.kh_internes_kennzeichen	VISIT_OCCURRENCE.visit_occurrence_id
FALL.aufnahmeanlass	VISIT_OCCURRENCE.visit_concept_id
FALL.aufnahmedatum	VISIT_OCCURRENCE.visit_start_date
FALL.entlassungsdatum	VISIT_OCCURRENCE.visit_end_date
"Visit derived from EHR record"	VISIT_OCCURRENCE.visit_type_concept_id
FALL.aufnahmeanlass	VISIT_OCCURRENCE.visit_source_value
FALL.aufnahmegrund	VISIT_OCCURRENCE.admitting_source_value
FALL.entlassungsgrund	VISIT_OCCURRENCE.discharge_to_source_value
CARE_SITE.care_site_id	VISIT_OCCURRENCE.care_site_name

Table 4: Insert a new care site.

source column	destination column
LABOR.concept_id	MEASUREMENT.measurement_concept_id
LABOR.timestamp	MEASUREMENT.measurement_date
LABOR.timestamp	MEASUREMENT.measurement_datetime
"Lab Result"	MEASUREMENT.measurement_type_concept_id
LABOR.value	MEASUREMENT.value_as_number
LABOR.concept_id	MEASUREMENT.measurement_source_concept_id
LABOR.LOINC	MEASUREMENT.measurement_source_value
LABOR.unit	MEASUREMENT.unit_source_value

Table 5: Insert measurement from LABOR.csv into MEASUREMENT.

source column	destination column
MESSUNGEN.concept_id	MEASUREMENT.measurement_concept_id
MESSUNGEN.timestamp	MEASUREMENT.measurement_date
MESSUNGEN.timestamp	MEASUREMENT.measurement_datetime
"From physical examination"	MEASUREMENT.measurement_type_concept_id
MESSUNGEN.concept_id	MEASUREMENT.measurement_source_concept_id
MESSUNGEN.LOINC	MEASUREMENT.measurement_source_value
MESSUNGEN.value	MEASUREMENT.value_as_number
MESSUNGEN.unit	MEASUREMENT.unit_source_value

Table 6: Insert measurement from MEASUREMENT.csv into MEASUREMENT.

source column	destination column
ICD.concept_id	OBSERVATION.observation_concept_id
FALL.aufnahmedatum	OBSERVATION.observation_date
"Observation recorded from EHR"	OBSERVATION.observation_type_concept_id
ICD.concept_id	OBSERVATION.observation_source_concept_id
ICD.icd_kode / ICD.sekundaer_kode	OBSERVATION.observation_source_value

Table 7: Insert observation from ICD.csv into the OBSERVATION.

source column	destination column
ICD.lokalisation	OBSERVATION.observation_concept_id
FALL.aufnahmedatum	OBSERVATION.observation_date
"Observation recorded from EHR"	OBSERVATION.observation_type_concept_id
ICD.lokalisation	OBSERVATION.observation_source_value
ICD.lokalisation	OBSERVATION.value_as_string

Table 8: Insert information about the location from ICD.csv to OBSERVATION.

source column	destination column
<table1>	FACT_RELATIONSHIP.domain_concept_id_1
<*table1*>.observation_id	FACT_RELATIONSHIP.fact_id_1
<table2>	FACT_RELATIONSHIP.domain_concept_id_2
<*table2*>.observation_id	FACT_RELATIONSHIP.fact_id_2
"Finding associated with/Associated with finding"	FACT_RELATIONSHIP.relationship_concept_id

Table 9: Connect two entries through a fact relation.

source column	destination column
ICD.diagnosensicherheit	OBSERVATION.observation_concept_id
FALL.aufnahmedatum	OBSERVATION.observation_date
"Observation recorded from EHR"	OBSERVATION.observation_type_concept_id
ICD.diagnosensicherheit	OBSERVATION.observation_source_value
ICD.diagnosensicherheit	OBSERVATION.value_as_string

Table 10: Insert information about the diagnostic reliability from ICD.csv to OBSERVATION.

source column	destination column
ICD.concept_id	CONDITION_OCCURRENCE.condition_concept_id
FALL.aufnahmedatum	CONDITION_OCCURRENCE.condition_start_date
FALL.aufnahmedatum	CONDITION_OCCURRENCE.condition_start_datetime
ICD.diagnoseart	CONDITION_OCCURRENCE.condition_type_concept_id
ICD.concept_id	CONDITION_OCCURRENCE.source_concept_id
ICD.icd_kode/ICD.sekundaer_kode	CONDITION_OCCURRENCE.source_value

Table 11: Insert conditions from ICD.csv to CONDITION_OCCURRENCE.

source column	destination column
OPS.concept_id	PROCEDURE_OCCURRENCE.procedure_occurrence_id
OPS.ops_datum	PROCEDURE_OCCURRENCE.procedure_date
OPS.ops_datum	PROCEDURE_OCCURRENCE.procedure_datetime
"Procedure recorded as diagnostic code"	PROCEDURE_OCCURRENCE.procedure_type_concept_id
OPS.ops_kode	PROCEDURE_OCCURRENCE.procedure_source_value
OPS.concept_id	PROCEDURE_OCCURRENCE.procedure_source_concept_id

Table 12: Insert procedures from OPS.csv into PROCEDURE_OCCURRENCE.

1.2 Decubitus Risk Prediction

The decubitus risk prediction is provided by a neural network, trained on patient data of the OMOP CDM. The network takes a number of clinical findings as input and predicts binary the risk of the patient developing a pressure ulcer in the near future. The specific findings used as input are determined by selecting the unique concept ids from the database which describe the patient's condition at the time a pressure ulcer is diagnosed. Therefore only findings are selected that could possibly be connected to the risk of developing a decubitus.

In the training process, old patient data is sampled at different moments pairing the information on the patient's condition given at this point and the binary label whether a pressure ulcer will be diagnosed within the next n dates contained in the database. N is a number to be configured prior to the training. The sampled data is split into a testset and a trainset. The network is trained only on the trainset, therefore the testset contains new unseen samples on which the network can be evaluated.

Once a neural network is trained on the trainset, validated on the testset and the classification threshold is selected, it can be loaded into the actual prediction pipeline. The Pipeline takes the latest patient data from the OMOP CDM as input and calculates the decubitus risk prediction. Afterwards, using gradient backpropagation, the Pipeline determines the importance of the different inputs to the calculated prediction. The most important clinical findings are filtered to reject meaningless Information and the top ten remaining facts are stored together with the prediction, to be displayed in the front-end.

1.3 Front-End

The front-end is a web application, running on a dedicated server, reachable only from the network of the medical institution. The patient data is protected by a login page, requiring a valid username and password. The data is extracted from the database, where the results of the decubitus risk detection pipeline are stored. The patient data is illustrated inside a table, where patients, potentially developing a decubitus are marked in red. The results can be downloaded and printed as a PDF file. Necessary parameters e.g. database connection settings can be configured as command line parameters on startup of the application.

2 Risk Analysis

As shown in Table 13 four major risk classes are considered. Firstly data loss or manipulation due to bugs, secondly the nonavailability of the service, thirdly attacks by an unauthorized person and lastly the risks of wrong patient treatment due to the software.

In the first risk, patient data in the OMOP database gets deleted or modified by a bug in the front-end. The severity of this risk is high because the correctness of the patient data is crucial and the data in the database in will not be checked on its correctness once it is inserted. The likelihood of this happening is low because the front-end never needs to write into the OMOP database and therefore the possibility of changing the data can be minimized by using only relevant read rights on database access.

In the second risk, patient data in the OMOP database gets deleted or modified by a bug in the ETL-Process. The severity of this risk is high because the correctness of the patient data is crucial and the data in the database in will not be checked on its correctness once it is inserted.

	Risk Description	Risk Likelihood	Risk Severity
1	Patient data gets deleted/modified by a bug in the front-end	low	high
2	Patient data gets deleted/modified by a bug in the ETL process	high	high
3	Patient data gets deleted/modified by a bug in the decubitus prediction back-end	low	high
4	Data of decubitus prediction gets deleted/modified by a bug in the front-end	medium	medium
5	Data of decubitus prediction gets deleted/modified by a bug in the ETL process	low	medium
6	Data of decubitus prediction gets deleted/modified by a bug in the decubitus prediction back-end	high	medium
7	The front-end is unavailable due to a dos attack	low	low
8	Unauthorized person tries to delete/manipulate data via the front-end	medium	high
9	Unauthorized person tries to get access to patient data via the front-end	medium	high
10	Patient is erroneously treated due to the decubitus prediction	middle	middle
11	Patient is erroneously not treated due to the decubitus prediction	middle	high

Table 13: Risk analysis

The likelihood of this happening is high because this processes task is to translate the data from CSV files to an OMOP Database, bugs and errors in the implementation are possible. Therefore the process needs to be tested to ensure the correctness of the implementation.

In the third risk, patient data in the OMOP database gets deleted or modified by a bug in the decubitus prediction back-end. The severity of this risk is high because the correctness of the patient data is crucial and the data in the database will not be checked on its correctness once it is inserted. The likelihood of this happening is low because the back-end never needs to write into the OMOP database and therefore the possibility of changing the data can be minimized by using only relevant read rights on database access.

In the fourth risk, data of the decubitus prediction gets deleted or modified by a bug in the front-end. The severity of this risk is medium because the correctness of the data is crucial to the functionality of the software, but the software is not crucial to the workflow in a hospital. The likelihood of this happening is medium because the front-end is responsible for the correct display of the information. Therefore unwanted modification of the data is possible and must be excluded by testing. The likelihood of data deletion is low because the front-end never needs to write on the data and therefore the possibility of deletion can be minimized by using only relevant read rights on access.

In the fifth risk, data of the decubitus prediction gets deleted or modified by a bug in the ETL-Process. The severity of this risk is medium because the correctness of the data is crucial to the functionality of the software, but the software is not crucial to the workflow in a hospital. The likelihood of this happening is low because the ETL-Process is independent of the decubitus detection and only severe bugs or errors would lead to data deletion/modification in an completely separate database.

In the sixth risk, data of the decubitus prediction gets deleted or modified by a bug in the decubitus prediction back-end. The severity of this risk is medium because the correctness of the data is crucial to the functionality of the software, but the software is not crucial to the workflow in a hospital. The likelihood of this happening is high because the prediction is generated in the back-end and errors are possible. Therefore testing and evaluation of the prediction is needed.

In the seventh risk, the front-end and thus the data on decubitus prediction is not available due to an attack overloading the providing back-end. The severity of this risk is low because the availability of this service is not crucial to the workflow in a hospital. This type of attack is one of the easiest ways to prevent the software's functionality, nevertheless, the likelihood is classified as low because the front-end will only be available within the hospital's intranet. Therefore an attacker would be required to act inside a controlled network where he would easily be spotted and the attack could be ended.

In the eighth risk, an unauthorized person tries to manipulate/delete data via the front-end. The severity of this risk is high because the correctness of the data is crucial and deliberate modification could lead to critical endangerment of patients. The likelihood is medium due to the fact that patient data front-ends are an easy target to manipulate the displayed data. Therefore the access to the front-end needs to be protected by a password authentication. Further, the front-end only provides read access on the displayed information and therefore no data in the database can be changed. Manipulations on the displayment of the data can be reverted by reloading the page.

In the ninth risk, an unauthorized person tries to access patient data via the front-end. The severity of this risk is high because patient data is highly confidential. The likelihood is medium due to the fact that patient data front-ends are the first target on acquirement of patient data. Therefore the access to the front-end needs to be protected by a password authentication. Further,

the front-end only provides selected patient data, e.g. the name, age, possibility of developing a decubitus and the top ten clinical findings leading to the decubitus prediction.

In the tenth risk, a patient gets erroneous treated to prevent the development of a decubitus. The severity of this risk is medium due to the possibility of medicamental treatment leading to unnecessarily side effects. However, drug administration should always be medically questioned. In addition, further measures, such as frequent repositioning, do not lead to any far-reaching impairment of the patient. The risk of an erroneously treatment only based on the software is possible but, as emphasized, the software should only be used as a guide, including verifying the prediction using the top ten clinical findings leading to the prediction. Therefore the likelihood is medium.

In the eleventh risk, a patient gets erroneously not treated leading to the development of a decubitus. The severity of this risk is high due to the far-reaching impairment of the patient. The likelihood is medium as the risk of a faulty refraining from treatment only based on the software is possible but, as emphasized, the software should only be used as a guide and the risk assessment should remain in the doctor's responsibility.

3 Quality Management

The software is developed according to the waterfall model.

- In the first step the task was analyzed and requirements for the software were documented.
- Secondly the software structure was designed. In this step, the software was split into the three subsections: ETL-Process, Prediction-Pipeline, and front-end. Interfaces, in form of database structures, were defined.
- In the third step, the sub-modules were implemented separately.
- In the fourth step, the sub-modules were firstly tested independently and afterward the complete pipeline was evaluated.
- The last step, the commissioning is prepared but not executed.

On completion of a step the fulfillment of the requirements of the preceding step were verified. On implementation the sub-modules were split into smaller tasks, which were implemented and tested on its own, followed by a system test verifying the correct operation of the task in the complete system.

4 Software Life-cycle Processes

4.1 Requirements

The following requirements for the software were identified:

- The software transforms CSV files containing standardized patient data according to §21 Paragraph 4 and 5 KHEntgG into an OMOP CDM.
- The software can be configured to perform the transformation periodically.

- The software provides a prediction based on patient data of the patients risk developing a pressure ulcer (decubitus).
- The prediction on the risk of decubitus development can be verified on provided reasons leading to the classification.
- The generated prediction can be displayed in a web browser.
- The access to the patient data via the web browser is protected by a password authentication.

4.2 Tests

The sub-modules of the software are tested independently due to the fact they only interact via defined interfaces in form of database entries. Therefore only the correctness from one interface to another is tested.

In addition, one complete run from CSV to front-end is performed to ensure the correct interaction via the interfaces.

4.2.1 ETL-Process

The different functions of the ETL-Process are tested by unit tests. Further, the complete sub-module is tested by inserting a set of CSV files into the OMOP CDM and checking the database content to contain the expected data.

4.2.2 Decubitus Risk Prediction

The decubitus risk prediction is based on a neural network, therefore the prediction can only be tested on test data. The F1-Score, recall, and precision are used to numerically evaluate the results. Further, the returned top ten clinical findings are evaluated via visual inspection of the results.

4.2.3 Front-End

The front-end was tested visually by displaying known data and checking the visualization for correctness.

4.3 Change Management

Updates for the subsections of the software can be deployed independently. Therefore the change management for each segment needs to be analyzed separately. This enables the fast deployment of bug fixes because only the affected sub-module needs to be updated, tested and deployed.

4.3.1 ETL-Process

If the ETL-Process is updated it will be validated on test data. Afterward it can be deployed. The deployment simply consists of the replacement of affected Python files. This can easily be done by using a versioning system like git. If a versioning system is used, the deployment only consists of the provider pushing the newest version and the customer pulling the newest version.

However, it should be noted that data already entered in the OMOP CDM will not be updated retroactively. Meaning, erroneously data in the database can only be fixed by deleting affected entries and inserting it as a set of CVS files into the updated ETL-Process.

4.3.2 Decubitus Risk Prediction

The update routine of the decubitus risk prediction can be split into two different cases.

In the first case, the update fixes a bug in the prediction pipeline. A bug is detected and fixed in the code, followed by testing of the sub-module. Afterward the new version can be deployed similar to the ETL-Process.

In the second case, the neural network (NN) is updated. Assuming the trainset is extended by new data the NN can be retrained on the new data. If the new network achieves better test results than the old version it can be deployed. For this purpose, the file that defines the NN is updated and deployed similar to the ETL-Process.

4.3.3 Front-End

If the front-end is updated, it will be validated through usability-tests[1], where the patient data is replaced with dummy data. If these tests succeed, the sub-module can be deployed by replacing the contained Python, CSS, JavaScript and HTML files. As the front-end only requires reading access to the database, two instances of different versions can run separately. Hence, if a newer version contains bugs or missing functionality, one can fall back to the previous version and report the issue to the developers. Therefore the front-end offers high availability.

5 Usability

5.1 Front-End

For the front-end, we chose a web application, written in Python. We did not offer the choice for a different approach in the questionnaire, as a web application offers the most flexibility. The admin is able to configure the desired information, and the actual users of the front-end do not need to concern about it. Another advantage is that no additional software has to be installed on the computers in the medical institution. Front-end users only need to open an arbitrary internet browser and navigate to the front-end page. Furthermore, in case the front-end needs to be updated, one has not to update the software on each computer.

5.1.1 Design

We offered various design options for the front-end in the questionnaire. In the end, the most popular design is illustrated in 2. Potential decubitus patients are marked in red, showing the user that he requires additional attention. If the user clicks on a red marked patient, a text appears, describing why DecubiTection decided that the patient may develop a decubitus. On the other hand, if the user clicks on a gray marked patient, a text appears that no risks concerning decubitus are found, but that there may be still a residual risk, as the detection algorithm is not perfectly accurate. In case, the user is confused by the front-end, we included a help button, illustrated in 3, which describes how to use the software.

[1] tests, where one or more developers validate the software through simply using it

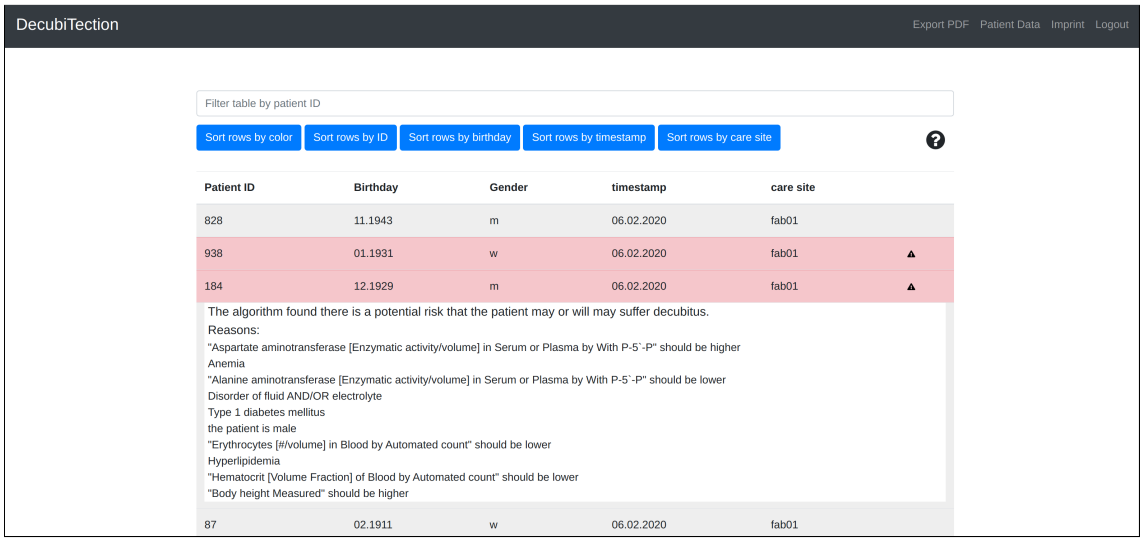

Figure 2: Most popular color theme of the front-end.

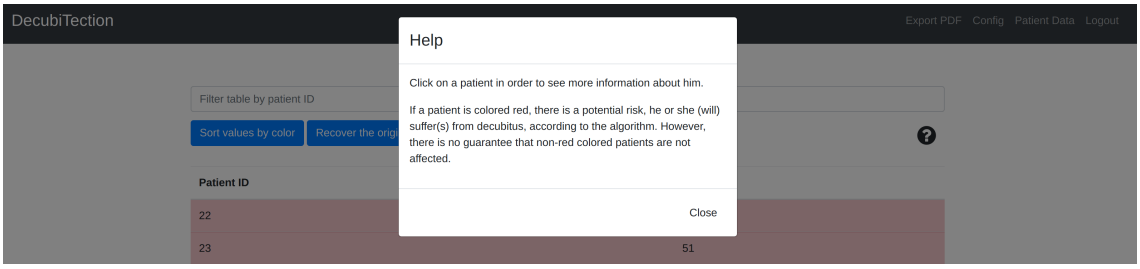

Figure 3: The help page of DecubiTection.

Though a web application brings many advantages, it contains a major disadvantage as well. Everyone inside the local network of the medical institution is able to access the front-end. We solved this by adding an authentication system, requiring valid credentials in order to access the front-end, depicted in 4. This decision matches the results of the questionnaire.

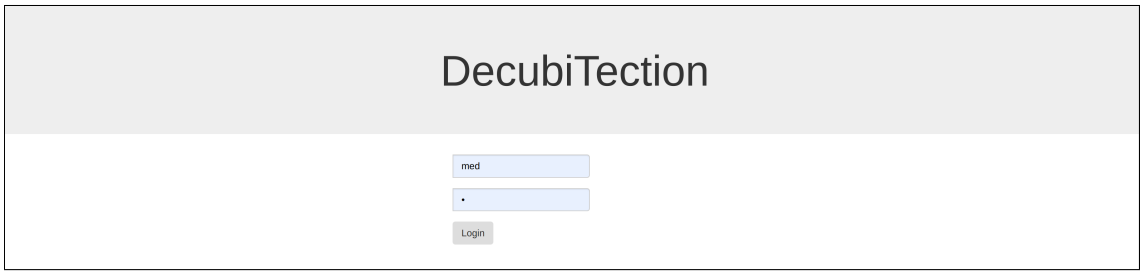

Figure 4: The login page of DecubiTection.

5.1.2 Content

The results of the questionnaire showed that as general information, at least the patient ID and the birthday of the patient are required. Additionally, we added the patient's gender and the care-site, the patient is currently located in. In the front-end, one can filter patients by their ID and sort them by background colors. We included the option to save the results as PDF file as well. Hence a user is able to print the results.

5.1.3 Configuration

There are different opinions, whether the front-end should include a configuration page e.g. for database credentials. We decided to not included such a page since it is not the front-end's users task to configure the software. In the current state, only an admin is able to configure the database settings and the credentials for authentication.

5.2 Back-End

The cron job for the ETL-Process and the detection algorithm needs to be configured, according to several parameters. According to the results of the questionnaire, it is sufficient to include these parameters through command line parameters, e.g. *python3 cron.py –db_port=1234 –db_host=localhost*. Furthermore, the questionnaire showed the necessity for the following parameters:

- cron interval
- for the database:
  - username
  - password
  - host
  - port

- location of the CSV files

Log files can be generated by redirecting the output of the software into a log-file.

6 Dependencies

6.1 ETL-Process

The submodule requires:

- python >= 3.6
- psycopg2 >= 2.8.4
- pandas >= 0.25.3
- numpy >= 1.17.4
- tqdm >= 4.38.0
- python-crontab >=2.4.0

6.2 Decubitus Risk Prediction

The submodule requires:

- python >= 3.6
- psycopg2 >= 2.8.4
- torch >= 1.4.0
- numpy >= 1.17.4
- tensorboardX >= 1.9
- matplotlib >= 3.2.0rc1
- tqdm >= 4.38.0

6.3 Front-End

The submodule requires:

- python >= 3.6
- psycopg2 >= 2.8.4
- flask >=1.1.1
- fpdf >=1.7.2