Summary of results and next steps

Characteristics of dataset

The dataset is built in Make_dataset.ipynb, in which SQL queries to the MIMIC-III database are done to retrieve creatinine measurements together with other interesting features : static information such as patient's age, diagnosis, and dynamic information such as arterial pressure. As regards dynamic variables that vary with time, we retrieve the last measurement that was done before the creatinine measurement, as well as the delay between these two measurements.

Important remark

During the Hackathon, we used the *labevents* table from MIMIC-III to retrieve all the lab results (including creatinine rates). However, following a discussion with other MIMIC users, the dates stored in *labevents* correspond to the date when a sample was taken, not when the results where known. For this reason, we decided to use exclusively values stored in the *chartevents*, where all the lab measurements are supposed to be reported. Indeed, *chartevents* contains another time variable (*storetime*) that is closest to the time at which the staff get the results from the lab.

Drawback : The dataset obtained from chartevents contains a lot less creatinine measurements.

Description of the dataset :

Basic statistics about the dataset can be found in Explore_dataset.ipynb.

Number of 24h-variations in creatinine rates : 36251

Number of features (including delays) : 61

List of features :

['creatinine' 'age' 'arterial_pressure_systolic'
'arterial_pressure_systolic_delay' 'arterial_pressure_diastolic'
'arterial_pressure_diastolic_delay' 'heart_rate' 'heart_rate_delay'
'temperature' 'temperature_delay' 'ph_blood' 'ph_blood_delay' 'ethnicity'
'diagnosis' 'gender' 'weight_daily' 'weight_daily_delay' 'urine_output' 'urine_output_delay'
'day_urine_output' 'day_urine_output_delay' 'scr' 'scr_delay' 'sodium'

'sodium_delay' 'potassium' 'potassium_delay' 'calcium' 'calcium_delay'
'phosphor' 'phosphor_delay' 'hemoglobine' 'hemoglobine_delay' 'uric_acid'
'uric_acid_delay' 'chloride' 'chloride_delay' 'platelet_count'
'platelet_count_delay' 'fibrinogen' 'fibrinogen_delay' 'urinary_sodium'
'urinary_sodium_delay' 'urinary_potassium' 'urinary_potassium_delay'
'urine_creatinin' 'urine_creatinin_delay' 'alkaline_phospatase'
'alkaline_phospatase_delay' 'total_protein_blood'
'total_protein_blood_delay' 'albumin' 'albumin_delay'
'total_protein_urine' 'total_protein_urine_delay' 'bilirubin'
'bilirubin_delay' 'c_reactive_protein' 'c_reactive_protein_delay'
'creatinine_yesterday' 'creatinine_before_yesterday']

The dataset is quite sparse (please refer to Explore_dataset.ipynb). For now, we didn't use any imputation method to replace missing values. A threshold was fixed (t=0.3) such that each feature with a rate of missing values > t is dropped. After that, each example with remaining missing values is also dropped.

Number of examples after dropping NAs and outliers : 27313

Number of features after dropping NAs : 15

List of features after dropping NAs :

['creatinine' 'age' 'arterial_pressure_systolic'
'arterial_pressure_systolic_delay' 'arterial_pressure_diastolic'
'arterial_pressure_diastolic_delay' 'heart_rate' 'heart_rate_delay'
'temperature' 'temperature_delay' 'ph_blood' 'ph_blood_delay' 'ethnicity'
'diagnosis' 'gender']

Characteristics of models

	Description	Number of features (after dummy encoding)	Number of examples in training set
linearSVM	linear Support Vector Machine	35	21850
logreg	logistic regression with l2 regularizer	35	21850

logreg_elasticnet	logistic regression with elasticnet regularizer	35	21850
XGBoost	Gradient boosted CARTs (Classification And Regression Trees)	35	21850
perceptron_1hl	Multilayer perceptron with 1 hidden layer	35	21850
perceptron_2hl	Multilayer perceptron with 1 hidden layer	35	21850
XGBoost_NAs	Gradient boosted CARTs with internal handling of missing values. This last model allows to keep missing values in the dataset and hence keep all features and examples.	66	35081

Please refer to Train_models.ipynb for further details.

Performances

Below are listed the performances obtained for each model on the test set. The performances that would be obtained for a random classifier are also reported for comparison. The metrics *sensitivity(inc)* and *specificity(dec)* respectively refer to the sensitivity when predicting an increase in creatinine rate, and to the specificity when predicting a decrease.

No cross-validation was performed to assess performances on the test set. However, to assess the typical variation in the performance metrics depending on the test set, this one was manually modified and the models re-trained. A variation of about 0.03 was observed for each metric when changing the test set.

	sensitivity(inc) (%)	**specificity(dec) (%)**
linearSVM	15	88
logreg	19	85
logreg_elasticnet	16	81

XGBoost	5	87
perceptron_1hl	1	85
perceptron_2hl	2	86
XGBoost_NAs	16	87
Random classifier	18	78

Discussion

Taking into consideration the 3% variation in performances when changing the test set, we can conclude that :

- The specificity when predicting a decrease is hardly better than the one obtained for a random classifier
- The sensitivity in increase is never better than the one of a random classifier ! This is really bad, as the increase in creatinine rate is precisely what we aim at predicting well.

Possible reasons for performing so badly :

- There are too few creatinine measurements available in *chartevents*
- The features we use are not predictive for variation in creatinine rates
- The class in which we are the most interested in ("increase"), is the one that is the rarest in the dataset. We may have too few examples in this class, or the metric we used to train the models (the mean accuracy over classes) may not be suited for predicting well the increase.

Next steps

Below are listed some hints to improve our results :

- **Check the building of the dataset** :
  - Check the SQL queries
  - Understand why we get a lot less creatinine measurements in the *chartevents* table compared to the *labevents* table
  - Understand why there are so many missing values, and why some features are empty
- **Use imputation methods to replace missing values**
- **Retrieve more features**
- **Use data augmentation to get a balanced dataset**

- **Try alternatives to the mean accuracy as loss function to optimize the models' parameters**