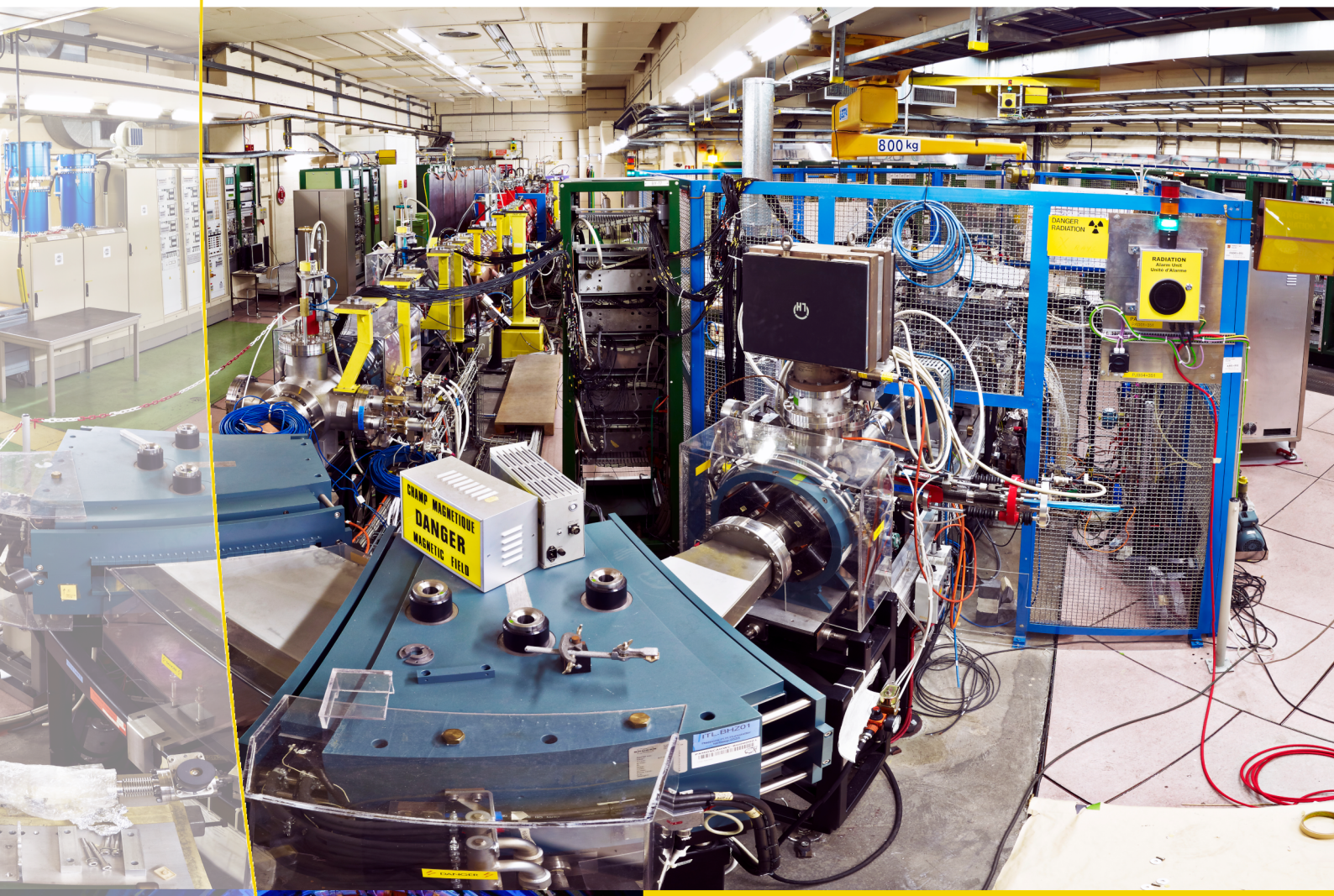
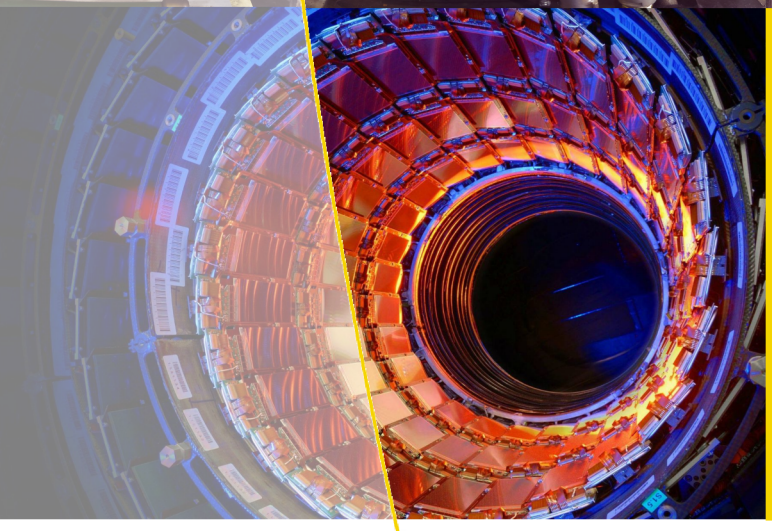
Faculty of Science
Mathematics Department

Universiteit Utrecht





Forecasting the LINAC3 Ion Beam Current

Report Written for: **Mathematics for Industy**

Written by:

Thomas **Breeman**
Renren **Gan**
Lucas **Hoogendijk**
Pieter **Knops**
Owen **Lynch**
Jeroen **Roberts**
Jelle **Soons**
Manousos **Theodosiou**
Jinyao **Tian**
Nils **Warsen**

Management Summary

In the last decade or so, the main attraction at CERN has been the Large Hadron Collider (LHC), which was indispensable to the discovery of the Higgs boson, as well as that of many other particles.[1] The LINAC3 is an integral part of the CERN accelerator complex, as it provides the initial ion beam used by the LHC and various other accelerators and experimental setups. The LINAC3, however, is an exceedingly complex machine, whose operation is dependent on a large number of input parameters. This results in a large variability of the intensity of the output ion beam. This report investigates how modern machine learning and prediction technologies can be implemented to devise a predictive system to warn the operators beforehand that the beam might decay, as well as to get better insight into what causes these decays. To tackle this problem, we have divided the group into three different subgroups. We have a group that applied time series analysis, a group that concerned itself with various data reduction and feature selection methods and a group that explored a neural network approach.
The main takeaway from our investigations is that *it is either not possible, or extremely difficult, to predict future beam decay based solely on observation of current beam state*. This conclusion was robust across models; even after applying sophisticated approaches based on current machine learning techniques, the ability of our models to forecast beam decay remained roughly on par with the persistence model (i.e. for every point in time the beam will stay the same as it is now). We have identified the following possible reasons for this:

- The fact that the operator is operating the machine could be removing patterns in the data; i.e. any "predictable" decay is already corrected for.
- Beam decay could be *truly chaotic* (i.e. actually impossible to predict on sufficiently large time scales).
- Beam decay could be *highly path-dependent*, and only successfully predicted through an analysis of *dynamics* of sensor measurement that intelligently aggregated data from multiple times.
- Beam decay could rely on some *hidden variables* that are not infer-able from current measurements.

However, although we were not able to produce a predictive model for beam decay, we have produced new insight into the beam decay problem that is potentially useful. From the feature selection and Granger causality research, we have identified subsets of columns that are worth looking into for future research, and additionally potential causes within the non-beam current columns. And finally, from the Ising model, we have given a model that flexibly segments the data.
We make the following recommendations for future research possibilities:

- The inability to predict the beam current can come from operators adjusting settings to prevent a beam decay, and disturbing the 'natural course' of the beam. An option is to reduce our dataset to times where the operator is not on duty.
- There can be some hidden variables that are not represented in the dataset but have a large influence on the beam current, e.d. the time since the last oven cleaning. Including extra variables, also by adding more sensors to the beam apparatus could help.
- Another possibility is to make a binary classification of the current into stable and unstable beams. Instead of predicting the beam current, one could predict this boolean.

Contents

1 Introduction

In the last decade or so, the main attraction at CERN has been the Large Hadron Collider (LHC), which was indispensable to the aforementioned discovery of the Higgs boson, as well as that of many other particles.[1] This report investigates how modern machine learning and prediction technologies can be implemented to improve the operation of the linear accelerator LINAC3, that supplies the LHC with heavy ions for its physics program, as displayed in the schematic below.

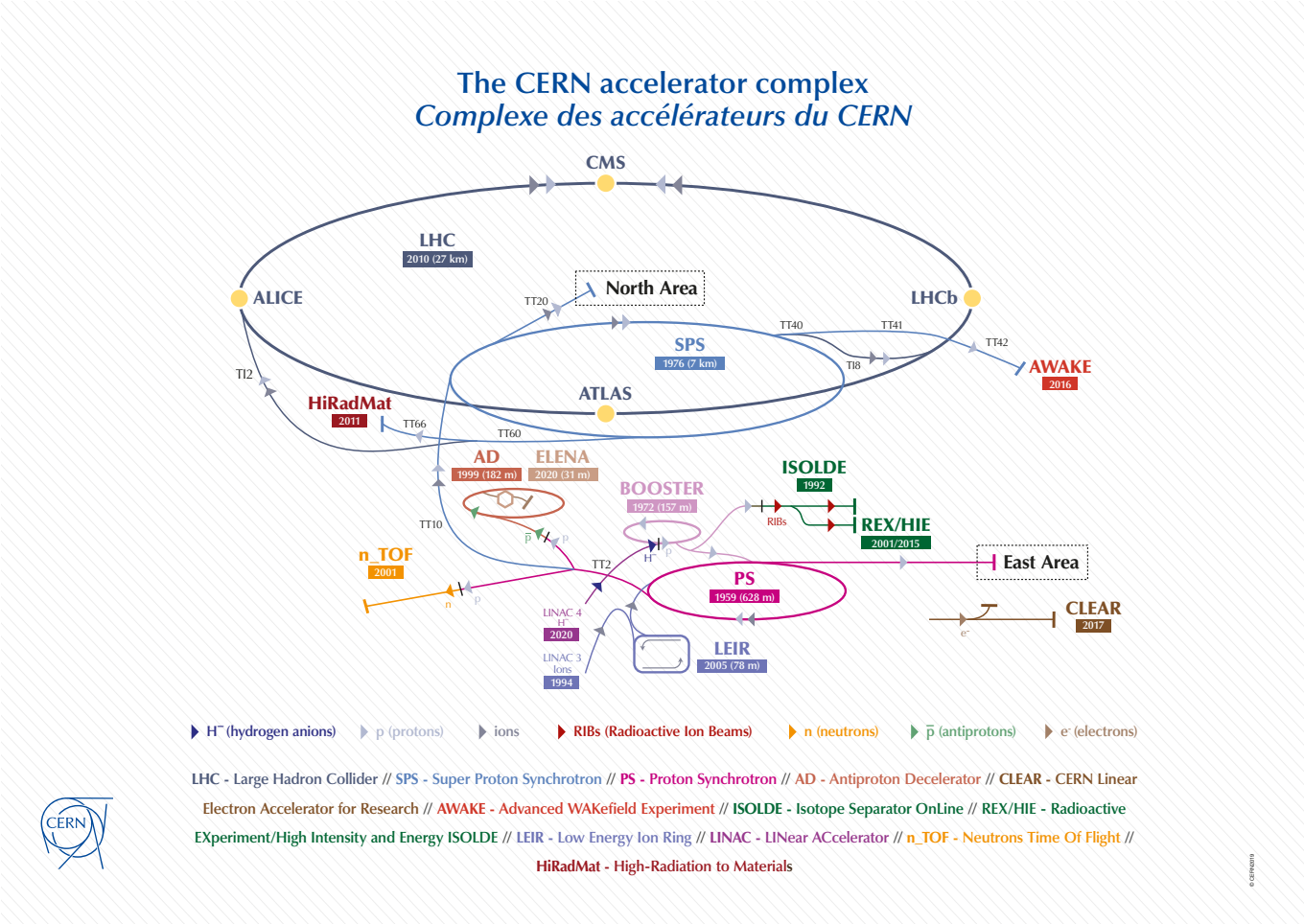


Figure 1: A schematic Of The Accelerator Complex

1.1 The LINAC3

The LINAC3 is an integral part of the CERN accelerator complex, as it provides the initial ion beam used by the LHC and various other accelerators and experimental setups. Initially, the LINAC3 was specifically designed to produce lead ions, which are heavier than proton beams and therefore have higher kinetic energies at the same speed. Later the LINAC3 was modified to also produce argon and xenon beams.
At the origin of the LINAC3 accelerator, lead atoms are stripped of some of their electrons (the remaining electrons are removed in a later stage), resulting in electrically charged lead ions. This charge allows for the ions to be manipulated by electromagnetic fields, so that they can be accelerated to some initial velocity and directed towards the Low Energy Ion Ring, where further preparation before injection into the LHC takes place. Because of this dependence of the LHC on the beam produced by the LINAC3, it is crucial that the latter operates as smooth and consistently as possible.
The LINAC3, however, is an exceedingly complex machine, whose operation is dependent on a large number of input parameters. This results in a large variability of the intensity of the output ion beam. The beam intensity regularly starts to decay without an easily detectable cause. The machine is operated by experts who are often able to prevent decay through a combination of intuition and experience, but as of yet there is no exhaustive solution which can be implemented whenever the beam starts to decay, or even a clear way to predict when the beam will start to decay. This last problem is the one we will deal with in this report, in which we search for appropriate machine learning and predictive techniques which can be implemented as an early warning system for beam decay.

1.2 Aim of the Project

It is our goal to devise a predictive system to warn the operators beforehand that the beam might decay, as well as to get better insight into what causes these decays. For this we will distinguish three levels of a predictive system.

1. Causality:
   This part will determine whether certain parameters caused the decay or could have a high contribution to the decay.

2. Prediction:
   This part will make an actual prediction on whether the beam will decay in a certain amount of time.

3. Decision support:
   This part will help with decision making. It will give an actual possible solution to make sure the beam will not decay.

All parts are closely related to one another. For example, understanding which parameters contributes the most to the decay, will help make a better prediction and can help with the decision making.
The aim of this project will be to explore various machine learning techniques in order to provide a better understanding of the parameters and the machine itself. This can be really useful for CERN and maybe in the future this can help to achieve a predictive system which outperforms the persistence model, which is currently surpassed only by very a small margin by all the models developed to this end.
To tackle this problem, we have divided the group into three different subgroups. We have a group that applied time series analysis, a group that concerned itself with Principal Component Analysis (PCA) and a group that explored a neural network approach. In the end we did not have the time or resources to delve into decision support. All the methods investigated by these subgroups will be discussed in the methods section.

2 Beam Data

For this project we initially received training data spanning from April to November 2018, with a sampling resolution of one data point per minute.
This data consisted of three categories. The first is the current of the outgoing beam, the second consists of the source settings adjustable by the operators, e.g. the power going into an oven, and the third is the data received from sensors in the LINAC3. This in total amounts to 305531 data points spread over 34 columns. A brief description of the data can be found in Appendix B.
Visual inspection of the data, as demonstrated in Section 2 leads to a different classification into categories of columns based on amount of activity. Some of the columns are highly dynamic, and have large amounts of noise. These columns include, for instance `IP.NSRCGEN:RFSAIREMAQNREV`, or `IP.NSRCGEN:BIASDISCAQNI`. Other columns are entirely static, like `IP.NSRCGEN:GASLOOP` or `IP.NSRCGEN:OVEN2AQNI`. And then there are some which are mixed; i.e. are mainly piece-wise constant, with rare spikes. These include `IP.NSRCGEN:SOURCEHTAQNV` and `IP.NSRCGEN:SOURCEINTERMEDIATEHTAQNI`.
The methods that we have used were primarily focused on the dynamic columns, and (of course) the beam current. We would have liked to have spent more time working to understand the rare spikes in the mainly constant data, but we did not ultimately have the time to do this. The spikes are especially tantalizing because they seem to be correlated across columns, for instance

`IP:NSRCGEN:SOURCEHTAQNV`, `IP:NSRCGEN:SOURCEINTERMEDIATEHTAQNV` and `IP:SOLEXT.ACQUISITION:CURRENT`.

Within the dynamic columns, there are clusters which are highly correlated. For instance, the columns

`IP:NSRCGEN:BIASDISCAQNV`, `IP:NSRCGEN:GAS2AQN`, `IP:NSRCGEN:GAS3AQN` and `IP:NSRCGEN:GASAQN`.

all seem to be slightly different measurements of the same underlying phenomenon.
Other correlated columns are

`IP:NSRCGEN:GASEXTRACTIONAQN` and `IP:NSRCGEN:GASSOURCEAQN`,

and

`IP:NSRCGEN:OVEN1AQNI`, `IP:NSRCGEN:OVEN1AQNP`, `IP:NSRCGEN:OVEN1AQNR` and `IP:NSRCGEN:OVEN1AQNV`.

Later on, we will do dimensionality reduction, and this visual categorization should serve as a useful sanity check for the validity/power of this reduction; ideally we should only be picking one column from each of these correlated categories.

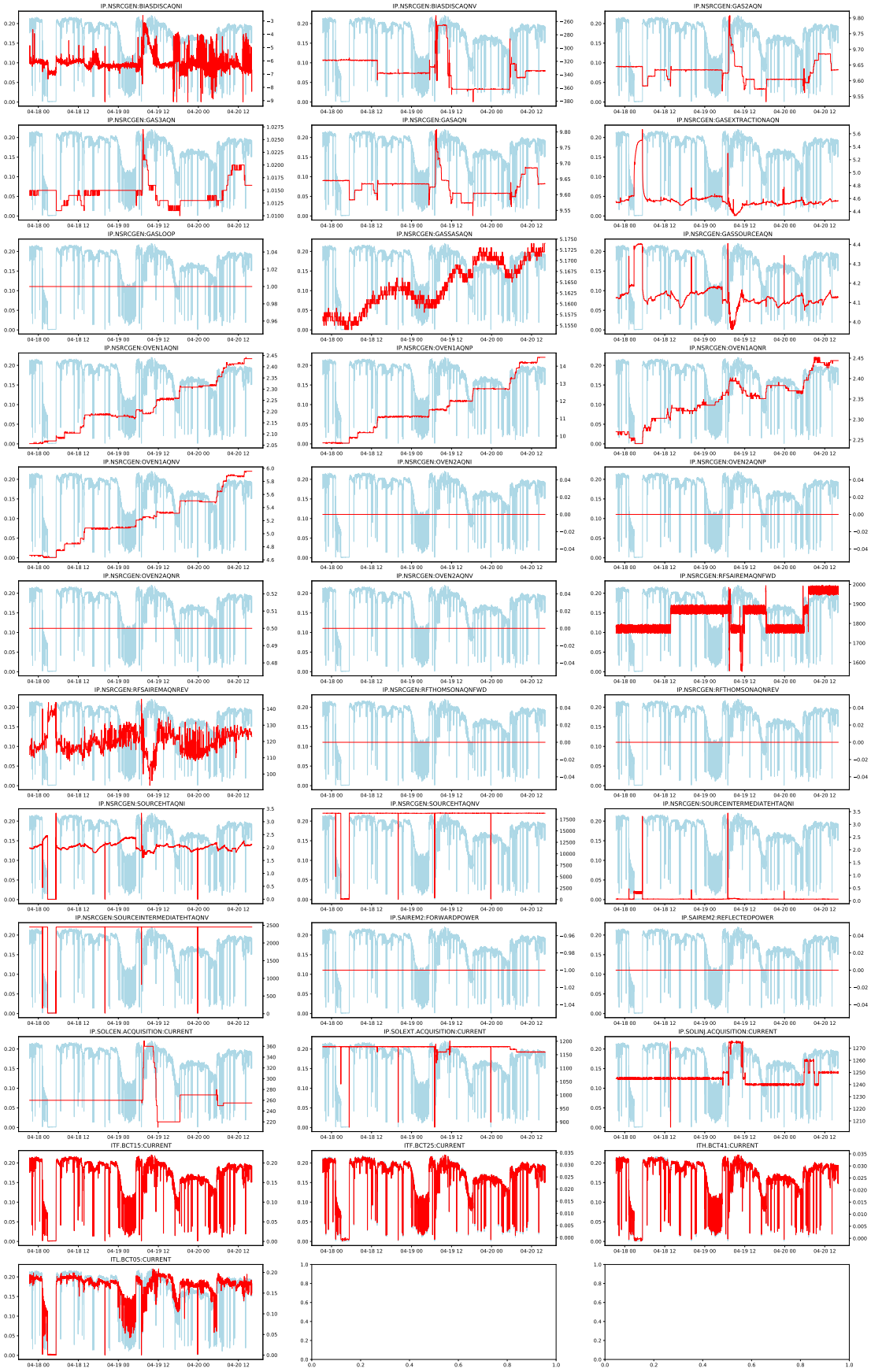


Figure 2: All Columns of the Data (Red), Plotted Against the Beam Current (Blue)

2.1 Preprocessing

The sensor data exhibits quite some stochastic "jittering", a form of noise in which the sensor value drops down significantly, for a short period of time. This jittering complicates training a Neural Network (NN) significantly [2, 3, 4]. Hence, we tried to remove such noise by using an exponential rolling average. The exponential rolling average computes the rolling average over a subset of the data, where the most recent data points are most important. In this way, we thought we could combat the jittering in the best way, without losing too much detail and without slowing the mean thus far that it has difficulties "snapping" onto new values. We did not want to lose information about the extent of the jittering. That is why we also calculated the exponential rolling variance averages of the data. This is done for each column in the dataset, except the time column, giving us 100 columns to choose from.

3 Methods

The methods that were investigated by the three subgroups fall into two broad categories. The first category was data description. In this category, we attempted to get a better understanding of the data by extracting features and seeing how well various models fit.
In the second category, we tried to make predictions on the data using a variety of machine learning techniques. These techniques included using some of the features found in the previous section to preprocess the data, which lead to modest improvements in learning performance.

3.1 Data Description

In this section, we describe methods that we used to get a better understanding of the LINAC3 behavior. Broadly speaking, these methods gave us a better understanding of the statistical properties of the data, i.e. correlations and lack of correlations.

3.1.1 Dimensionality Reduction

In Section 2, visual inspection of the data revealed that many of the time series are highly correlated, and many of the time series are highly static. This implies that we may be able to reduce our number of features by a great deal without sacrificing model accuracy. Making this reduction also tells us something about what is important in the data.
We chose to reduce dimensionality by eliminating features. We did this via an ensemble of three methods.

1. The LASSO, which fits a linear model and tries to make as many coefficients 0 as possible. The LASSO method regularizes model parameters by shrinking the regression coefficients, reducing some of them to zero. The feature selection phase occurs after the shrinkage, where every non-zero value is selected to be used in the model. The amount of penalization α was chosen by cross validation. The LASSO is initially due to [5], and we used the implementation in [6].

2. Recursive feature elimination (RFE) applied to a random forest model. RFE is a general strategory for removing unnecessary features from a model. It attempts to remove as many features as it can without degrading model performance too much. The theory of RFE is laid out in [7], using the random forest model from [8]. The implementation we used was the one in scikit-learn [6], using techniques from [9].

3. RFE applied to a random forest model with gradient boosting. The same description above applied, only using gradient boosting instead of random forests, which is described in [10].

In each of these methods, the features are evaluated based on their utility in predicting the beam current 15 minutes in advance; more discussion on the relation between supervised learning and time series will be had in a later section. We removed a feature if all of these models reported that the feature was unnecessary. This reduced feature set both told us something interesting about the data itself, and also helped us develop an improved supervised learning model later on.

3.1.2 Beam Classification

We conducted an analysis of the beam current based on the assumption that the beam has two essential states: "steady" and "failing." These states are characterized by a variety of features, but the most important one is low beam current over a sustained period of time. Corresponding to this, we made a classification of beams based on local averages of the beam current. Using this classification, we explored the distribution of how long the beam remained steady.

We also implemented a more sophisticated model for classification of "steady" versus "failing" beams, that captures more directly the assumption that beams are "steady" or "failing" for long periods of time. This assumption was implemented by a statistical mechanics model which assigns high energy to proposed classifications which frequently change from "steady" to "failing", and low energy to proposed classification which don't jump between states very often. The inspiration for this model is due to [11], and the implementation in terms of Gibbs sampling is taken from [12].

3.1.3 Traditional Time Series Models

We then attempted to model the beam current using an autoregressive model. This gives a prediction for X_t as a linear combination of $X_{t-1}, \ldots, X_{t-k}$. We investigated how well the autoregressive model described the data, and whether or not it could be used for prediction. This is the most traditional approach to time series modelling, and is laid out in [13].

3.1.4 Granger Causality

Finally, we performed a Granger causality analysis [14] of the different exogenous variables on the main beam current. This analysis measures whether an autoregressive linear model involving a given exogenous variable *and* the main beam current makes better predictions than a model just involving the main beam current.

3.2 Supervised Learning

The basic task for supervised learning is to reconstruct either *continuous* or *discrete* labels from *labeled* data. There are many off-the-shelf algorithms for supervised learning available in libraries such as scikit-learn [6]. The assumption behind supervised learning is that one is taking independent and identically distributed (i.i.d.) samples from a fixed distribution for two variables, X and Y. In classification, Y takes values in a finite set, and in regression, Y takes values in $\mathbb{R}$. Typically, X takes values in $\mathbb{R}^n$. The objective of the supervised learning is to find an estimator $\hat{y}(X)$ such that $\mathbb{E}[d(\hat{y}(X), y)]$ is minimized, where d is some distance measure.
Supervised learning takes in data in "tabular form", where X is a $d \times n$ matrix (with d the number of features) and y is a length-n vector. Reducing a time series prediction task to "tabular form" requires non-trivial decision making [15], and we will discuss the various approaches that we took in the sections below.

3.2.1 General Models

The most basic approach uses the value of all of the time series at time t as X, and the value of the beam current at time $t+T$ as y. More precisely, if $B_{t,1:n}$ is a *multivariate* time series, and we are interested in predicting $B_{t+T,1}$, then we sample t randomly, and let $X = B_{t,1:n}$ and $y = B_{t+T,1}$. This approach relies on the following assumptions.

1. We can pick times t so that our collection of samples is roughly i.i.d. (or more generally, *exchangable*).
2. The time series is Markovian, so that $B_{t+T,1}$ is independent of $B_{t-k,j}$ for $k \geq 1$, given $B_{t,1:n}$.

In this approach, the *persistence model* sets $\hat{B}_{t+T,1} = B_{t,1}$.
We assumed that we had enough data so that overfitting due to non-i.i.d. samples would not be a problem. If we had suspiciously good predictions, we would have worried about whether the model had simply learned every dip of the data, but, as we will discuss later, this was not the case.
A more sophisticated approach is to learn on a transformation of the raw time series data, as discussed in the Data section. For instance, we might use a *running mean*. Any of the previous supervised learning techniques can be applied to this "smoothed" data. Moreover, $\hat{B}_{t,1:n}$ only relies on data up to time t, so it is valid for predicting. We can also transform the y data; if we are interested in just predicting general trends, it might be useful to smooth it out. We do this with a Gaussian convolution. Finally, in computing the running mean, we might have lost important information coming from the variance; i.e. a noisy beam could be an indicator of a future decay, even if the mean is high. We account for this by looking at a "running variance" in addition to the running mean.
We ran a variety of off-the-shelf algorithms on different combinations of these preprocessing techniques, to get an idea of what kind of baseline performance simple models would do. These off-the-shelf algorithms all come from scikit-learn [6], and we list them along with their name in that library.

1. Linear Regression,
   `sklearn.linear_model.Linear`
2. Ridge Regression,
   `sklearn.linear_model.Ridge`

3. Perceptron Regression,
   `sklearn.neural_network.MLPRegressor`
4. Gradient Boosted Regression,
   `sklearn.ensemble.HistGradientBoostingRegressor`
5. Support Vector Regression,
   `sklearn.svm.SVR`

3.2.2 Focused Model: Neural Net

After the off-the-shelf algorithms, we tried again with a custom neural net architecture, only looking at the features that the data description team identified as most important, and augmenting with running means, variances, and past data. This more advanced model was implemented using Keras [16], and we will talk more about the details in the results section, where we discuss how we modified the architecture based on the ouputs that we were getting.

3.2.3 Focused Model: XGBoost

Among the machine learning methods used in practice, gradient tree boosting is one technique that shines in many applications. Tree boosting has been shown to give state-of-the-art results on many standard classification benchmarks. XGBoost stands for "Extreme Gradient Boosting"[1]. In brief, boosting uses sequences of decision trees that seek to reduce the residuals of the prior tree. In other words, each new tree uses the residual of the prior tree as the target variable for the current tree. In doing so, for each new tree, greater focus is given to the larger errors of the prior trees. After enough trees are created, the sum of all tree predictions is calculated to generate the final predicted value.
There are three of the primary enhancements of XGBoost:

- Two forms of regularization which are meant to reduce the potential of overfitting prediction.
  - L1 (LASSO) regularization, which will push feature weights to zero and is represented by the alpha parameter.
  - L2 (ridge) regularization, which will push feature weights asymptotically to zero and is represented by the lambda parameter.
- Gamma parameter: control the complexity of a given tree. The gamma parameter is used to set the minimum reduction in loss required to make a further split on a leaf node. XGBoost will use the gamma parameter in its pruning steps. It will grow the trees to a specified level and then use the gamma parameter in determining which leaf nodes to remove.
- Scalability and parallel distributed computing. The most important factor behind the success of XGBoost is its scalability in all scenarios. The system runs more than ten times faster than existing popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings. Parallel and distributed computing makes learning faster which enables quicker model exploration. More importantly, XGBoost exploits out-of-core computation and enables data scientists to process hundred millions of examples on a desktop.

4 Results

4.1 Data Description

4.1.1 Dimensionality Reduction

Recall that we designed a system of three feature eliminators, the LASSO, RFE with a random forest, and RFE with gradient boosting, and we eliminated a feature if and only if it was eliminated in all three of these models. The result of this process was that only the features in the following list were retained.

- `Time`
- `IP.NSRCGEN:GAS3AQN`
- `IP.NSRCGEN:GASAQN`
- `IP.NSRCGEN:OVEN1AQNR`
- `IP.NSRCGEN:RFSAIREMAQNFWD`
- `IP.NSRCGEN:SOURCEHTAQNI`
- `IP.NSRCGEN:SOURCEHTAQNV`
- `IP.SAIREM2:FORWARDPOWER`
- `IP.SOLEXT.ACQUISITION:CURRENT`

[1]Gradient tree boosting is also known as gradient boosting machine (GBM) or gradient boosted regression tree (GBRT)

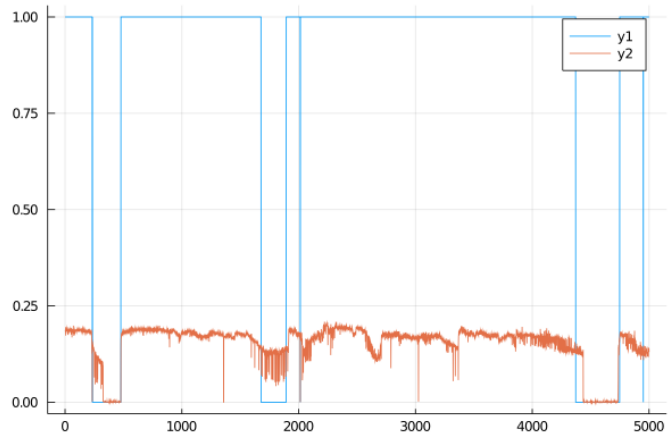
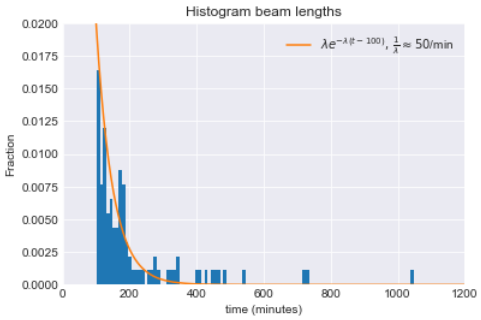


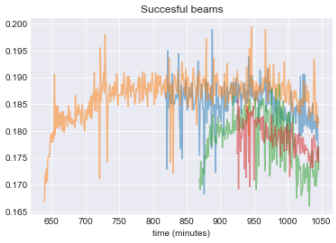
Figure 4: Ising Model Classification of Good/Bad Beams

4.1.2 Beam Classification

We used exponentially averaged current data I (`ITL.BCT05:CURRENT`). We selected from the data only the stable beam periods: time intervals of at least 100-minute-long periods where $I > 165\mu A$. This yields 97 successful beams. In Figure 3(b) a couple of these beams are plotted, while in (a) the histogram of beam lengths is shown. It appears that the beam lengths are roughly exponentially distributed. We fit $\lambda e^{-\lambda(t-100)}$ to the histogram, yielding $\lambda \approx 0.02\text{min}^{-1}$. So on average you expect the current to be above $165\mu A$ for an average time of $\frac{1}{\lambda} \approx 50\text{min}$.



(a) Histogram of successful beam lengths.



(b) An example of a couple of successful beams, all ending at 1050 minutes (length of longest beam).

Figure 3: The exponentially averaged data split into successful beam intervals.

That these lengths are roughly exponentially distributed suggests that we can treat the current as some stochastic process I_t where t is time. We made the assumption that each individual successful beam is a realization of this same process I_t. This hypothesis was important in later modelling with traditional time series models. In this section, we also discuss the more sophisticated model for classifying beams based on statistical mechanics. We then sampled from this system using Gibbs sampling. The result is depicted in Figure 4.

We see that, without applying any smoothing, the system was able to successfully detect that small deviations are not bad beams, but large deviations are. However, the Gibbs sampling procedure needed to compute this is fairly expensive, and might not scale very well to larger time windows without a rethink of our algorithm.

Further directions that we could have taken with this method would include using a more sophisticated "local magnetization" function to bias the Ising model towards good/bad beams, which perhaps could take into account more of the other time series. Additionally, once the good/bad beam data had been well-classified by this classification scheme, we could use machine learning for classification, rather than regression. However, we have not had time to attempt either of these approaches yet, which we recommend for future research.

4.1.3 Traditional Time Series Models

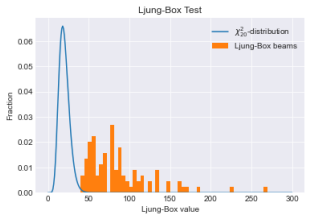
Looking at Figure 3(b) there is obviously some dependence between current values at different times. As a first step we took the differenced current for each beam:

$$\nabla I_t = I_{t+1} - I_t. \tag{1}$$

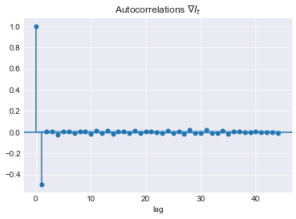
Looking at Figure 6(a) we saw that ∇I_t has no clear trends, and we assumed the series was stationary. We tested the differenced process ∇I_t for dependence between values of different times. For each beam we calculated the *Ljung-Box* statistic:

$$L = \sum_{h=1}^{k} \frac{n(n+2)}{n-h} \hat{\rho}(h) \tag{2}$$

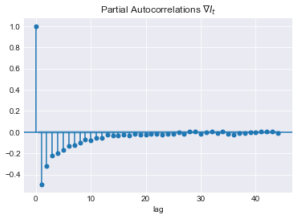
where n is the length of the series ∇I_t of each beam and $\hat{\rho}$ the observed auto-correlation at lag h for the beam. If the series were independent, then $L \sim \chi^2_k$ (a chi-squared distribution with k degrees of freedom), which is not case, see Figure 5(a). Next, we pasted ∇I_t of all beams together in order, for a new series Y_t and computed the autocorrelations and partial autocorrelations of this large series, see Figure 5(b) and (c). Note that Y_t is not exactly the same as the stochastic process ∇I_t, but the 96 artificial transitions in Y_t by pasting the beams together will only have a marginal effect on the whole series Y_t of length 20165 min.



(a) Histogram of Ljung-Box statistic of the beams for $k = 20$, together with χ^2_{20}-distribution.



(b) Autocorrelations of Y_t.



(c) Partial autocorrelations of Y_t.

Figure 5: Testing the independence of the differenced current.

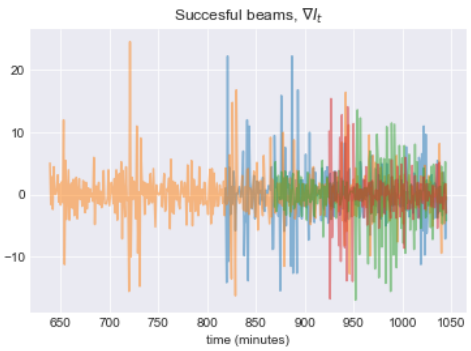
From Figure 5 we concluded that there is obvious dependence within the differenced current, as the Ljung-Box test indicates. Moreover, the autocorrelations show that Y_{t+h} and Y_t are only correlated for lag $h = 1$, while the partial correlations suggest that the impact of an extreme Y_t on next value Y_{t+h} exponentially decays as h grows. Since the autocorrelations are abruptly zero for lags greater than one, we suspect a moving average process of degree one (MA(1)):

$$Y_t = Z_t + \phi Z_{t-1} \tag{3}$$

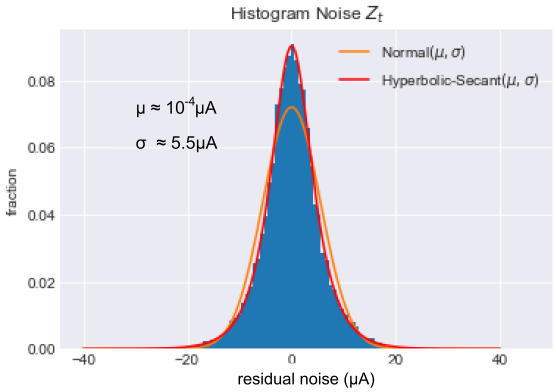
where ϕ is a constant coefficient, and Z_t a white noise sequence. Note that this way Y_{t+h} only depends on Z_{t+h} and Z_{t+h-1}, while Y_t only depends on Z_t and Z_{t-1}. Hence Y_{t+h} and Y_t are indeed uncorrelated for $h > 1$.

The model $Y_t = \alpha + Z_t + \phi Z_{t-1}$ was fitted to the series Y_t. This yielded $\alpha = 0.0061 \pm 0.004 \mu$A and $\phi = -0.8876 \pm 0.003$. Note that $\alpha \approx 0\mu$A. A non-zero α would have suggested that the series Y_t is more likely to be positive (or negative), and hence there would have been some drift in the combined (pasted next to each other) currents I_t of the beams. This violates our assumptions that all beams are just realizations of the same process, and that it does not matter at what time of the year the beam takes place. Hence we took $\alpha = 0\mu$A. We also copied $\phi = -0.9$ from the fit. Therefore our model for the differenced current:

$$Y_t = Z_t - 0.9 Z_{t-1}. \tag{4}$$



(a) The differenced currents ∇I_t in μA for the beams already shown in Figure 3(a).



(b) Histogram of the noise Z_t obtained from the fit to Y_t.

Figure 6

From the MA(1)-fit we also obtained the series Z_t as noise. The autocorrelations as well as the partial autocorrelations both did not significantly differ from zero for lags $h > 0$, so it seems Z_t is an independent white noise series. As an extra check we also computed the *Hurst*-exponent H of series Z_t. The Hurst exponent $0 \leq H \leq 1$ indicates whether a series is persistent ($H > 0.5$), anti-persistent ($H < 0.5$) or a Brownian time-series ($H \approx 0.5$). For Z_t we find $H = 0.43$, which is quite close to 0.5, so we can safely take Z_t to be a white noise series.

In Figure 6(b) the histogram of Z_t is plotted, which shows that Z_t has a quite high kurtosis. The kurtosis of Z_t is 4.57, so we have an excess kurtosis of 4.57-3 = 1.57. This can be an indication that Z_t is the result of a GARCH-process, which is also suggested by some slight volatility-clustering visible in Figure 6(a). However, if this were the case we would also observe dependence in the squared time-series Z_t^2. Since Z_t^2 has (partial) auto-correlations close to zero for lags $h > 0$, we abandon the idea that Z_t is the result of a GARCH-process.

A second approach to model Z_t was to fit some distribution to the histogram of Z_t in Figure 6(b). The series Z_t has average $\mu = 10^{-4}\mu$A$\approx 0\mu$A, and standard variation $\sigma \approx 5.5\mu$A. We laid the histogram over a normal distribution $\mathcal{N}(\mu, \sigma)$, which did not seem to be a good fit. However, the hyperbolic secant distribution ($\mathcal{HS}(\mu, \sigma)$) fit the histogram of Z_t quite well! The pdf of the hyperbolic secant is defined as:

$$f(x; \mu, \sigma) = \frac{1}{2\sigma} \frac{1}{\cosh(\frac{\pi}{2}\frac{x-\mu}{\sigma})}. \tag{5}$$

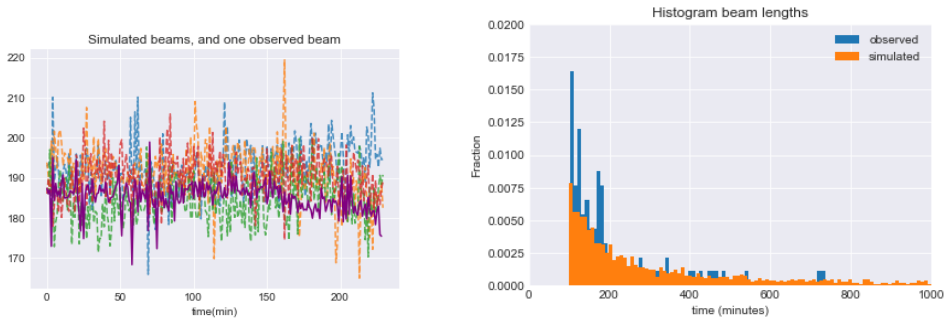
So we took that $Z_t \sim \mathcal{HS}(0\mu A; 5.5\mu A)$.

Finally, our stochastic model for the current I_t was completed. We derived that $\nabla I_t = Z_t - 0.9Z_{t-1}$ where $Z_t \sim \mathcal{HS}(0; 5.5)$, so:

$$I_t = I_{t-1} + Z_t - 0.9Z_{t-1}. \tag{6}$$

We tested this model by comparing simulations to observed data. In Figure 7(a) four beams are simulated beams and one observed. The simulated beams behaved somewhat more erratically than the actual observed beam. To make a better comparison between observation and simulation we simulated 10000 beams, all with starting value $I_0 = 178\mu A$ (same as the average starting value of the 97 successful beams), and let them run until the current dips below $165\mu A$. Then we took out any beams shorter than 100 minutes, just as we had done with the observed beams. The histograms of the observed and simulated beam lengths are in Figure 7(b). It can be seen that the stochastic model underestimates the number of beams shorter than 3 hours.

All in all, the stochastic model is not bad but also not that great at reproducing observed data. This might be the best one could expect given that we solely modelled the current `ITL.BCT05:CURRENT` without including any other observed parameters.



(a) Four simulated beams using Equation (6) and one observed beam in purple, all with same initial value I_0.

(b) Histogram of the lengths of 10000 simulated beams, all with $I_0 = 178\mu A$, together with the histogram of observed beam lengths (same as in Figure 3(a)).

Figure 7: Results from simulation experiments using Equation (6).

Lastly, we considered whether the stochastic model can be used to make predictions 15, 30 and 60 minutes ahead. The best linear predictor, given in Equation (6), for I_{t+h} given that we know $I_0, I_1, \ldots, I_t$ and $Z_{-1}, Z_0, \ldots, Z_t$ is:

$$\Pi I_{t+h} = I_t - 0.9Z_t \tag{7}$$

since white noise Z_t cannot be predicted based on past data the best prediction for Z_{t+1} is just zero. The theoretical prediction error can be derived by recursively applying Equation (6):

$$I_{t+h} - \Pi I_{t+h} = Z_{t+h} + 0.1 \sum_{i=1}^{h-1} Z_{t+i}. \tag{8}$$

Since the hyperbolic secant distribution is a scale-free distribution, we have

$$\begin{aligned} & Z_{t+h} + 0.1 \sum_{i=1}^{h-1} Z_{t+i} \\ & \sim \mathcal{HS}\left(\mu + 0.1(h-1)\mu, \sqrt{\sigma^2 + 0.1(h-1)\sigma^2}\right) \\ & = \mathcal{HS}\left(0, 5.5\sqrt{1 + 0.1(h-1)}\right). \end{aligned}$$

So the theoretical prediction error for only 15 minutes ahead ($h = 15$) already has a variance of roughly 5.8μA, which is not that large compared to the average starting value of 178μA: only roughly 3.3%. However, compared to the difference with the 'good beam threshold' of 165μA a beam only needs to decay by about 10μA to classify as bad beam. Compared to this the variance is quite large. Hence, the stochastic model is not suitable to make predictions.

4.1.4 Granger Causality

We have applied the Granger causality tests to only a subset of the time series for the beam data, because several of the time series did not change frequently enough to provide sufficient-rank matrices for least-squares fitting. What we see in Figure 8 is the "f-val" coefficient for each of the columns that change frequently enough, with an auto-regressive model using 15 values back in the past. Roughly speaking, the "f-val" measures how much more of the variance can be explained by the model with both the auxiliary time series and the main time series, as opposed to just the model with the main time series.
We also applied the Granger causality test to every single pair of dynamic columns, to get the graph in Figure 9. The plot of all Granger causalities reveals that the oven power has a strong causal influence on the gas pressure, which makes a lot of sense, given that oven power is a setting and gas pressure is an acquisition which could plausibly depend on oven power. We suspect that other Granger causality results could make similar sense given more understanding of the machine, but we are not familiar enough with the physics behind LINAC3, to make sense of the data in this way.

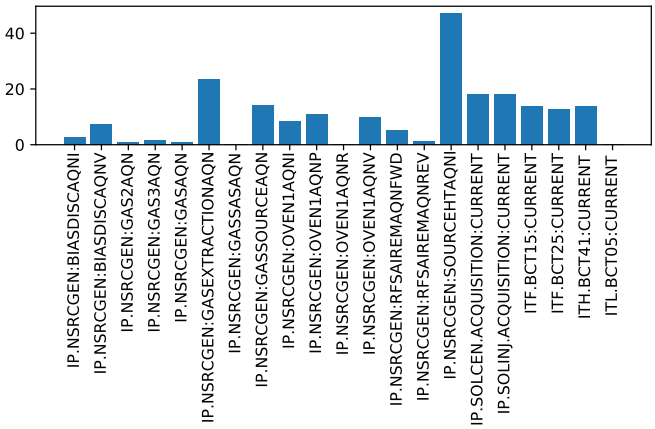


Figure 8: Granger Causality f-vals for Subset of Columns

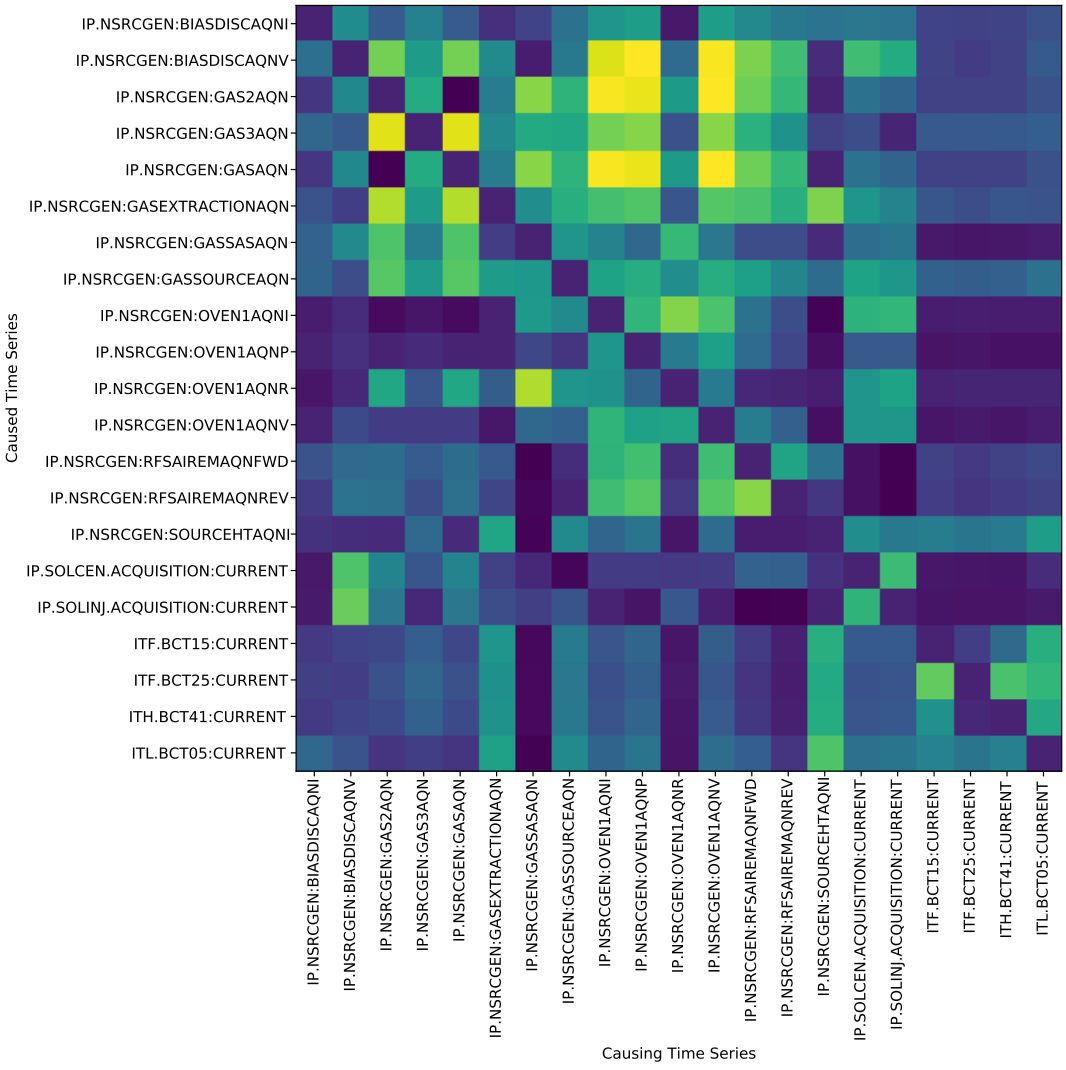


Figure 9: Log of Pairwise Granger Causality f-vals for all Columns, where lighter hues represent higher causality

4.2 Supervised Learning

4.2.1 General Models

We discussed our procedure for the off-the-shelf models used for regression in this section, and the results are summarized in Table 1. We combinatorially tried all combinations of preprocessing for X, preprocessing for y, and algorithm used. We see that a score around 0.90 is typical, and 0.96 is in the high end. However, this high end is achieved by smoothing the y, which would of course lead to a lower prediction error. Ultimately, the main conclusion is that it doesn't seem like any of the models work any better than others.

	Xtype	ytype	model	score
0	RAW	RAW	linear	0.867271
1	RAW	RAW	ridge	0.867267
2	RAW	RAW	neural	0.860908
3	RAW	RAW	gradient_boost	0.907159
4	RAW	RAW	svr	-0.318278
5	RAW	SMOOTH	linear	0.916850
6	RAW	SMOOTH	ridge	0.916845
7	RAW	SMOOTH	neural	0.907062
8	RAW	SMOOTH	gradient_boost	0.956007
9	RAW	SMOOTH	svr	-0.346541
10	MEANS	RAW	linear	0.874358
11	MEANS	RAW	ridge	0.874346
12	MEANS	RAW	neural	0.882958
13	MEANS	RAW	gradient_boost	0.913442
14	MEANS	RAW	svr	-0.306423
15	MEANS	SMOOTH	linear	0.923661
16	MEANS	SMOOTH	ridge	0.923649
17	MEANS	SMOOTH	neural	0.891227
18	MEANS	SMOOTH	gradient_boost	0.961840
19	MEANS	SMOOTH	svr	-0.346541
20	MEANSVAR	RAW	linear	0.877514
21	MEANSVAR	RAW	ridge	0.877494
22	MEANSVAR	RAW	neural	0.889851
23	MEANSVAR	RAW	gradient_boost	0.918245
24	MEANSVAR	RAW	svr	-0.324810
25	MEANSVAR	SMOOTH	linear	0.926728
26	MEANSVAR	SMOOTH	ridge	0.926721
27	MEANSVAR	SMOOTH	neural	0.942634
28	MEANSVAR	SMOOTH	gradient_boost	0.966506
29	MEANSVAR	SMOOTH	svr	-0.346541

Table 1: Prediction Results For Tabular Regressions

4.2.2 Focused Models: Neural Net

After the wide variety of approaches in Table 1, we attempted to give a more focused approach with a specialized neural net architecture. In this part, we chose to use the rolling means of the data corresponding to the 9 columns discussed in Section 4.1.1 and the 7 most important columns proposed by CERN, which are stated in the Appendix. Since we are interested in predicting the future, we did not feed the NN only the current state, but also some features of the past state, i.e., we assumed that the system is not entirely Markovian. We did this by feeding it two sets of delayed values. E.g., for the 60-minute forecast, we fed it the current values, all values but with a 60-minute delay, and all values but with a 120-minute delay.

In this way, we had 28 (or 21) input features and we used 3 hidden layers, with a total of 70 ReLu-nodes. Never have we noticed that increasing the number of layers or nodes did increase performance. For the 60-minute forecast, both sets of columns performed in terms of R^2 and RMSE a little better than the persistence model, although not significantly. For the 9 column set, it is visible that this model misses the crashes of the beam (there are many points on the horizontal line for a predicted current of around 0.165mA).

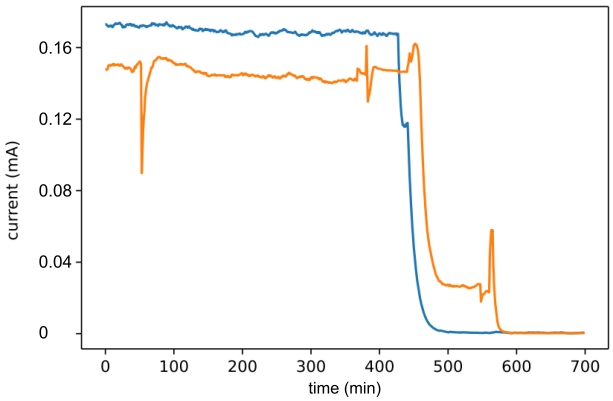


Figure 12: The prediction of the NN (orange) missed the downfall that occurred after a time of 400 minutes.

We tried a few things to counter this persistence model, which misses current downfalls.
One thing that we realised is that, in the data, the beam is in the stable state. It then might be totally reasonable that the neural net learns persistence. With this in mind, we decided to skew the data a bit so that it would contain less stable beams. For the case where we tried to have a ratio of about 1:1 between good beams and bad beams, we had the following results.
One attempt to try to combat the persistence model was by removing persistent beam regions, in other words only feeding it the small regions in which the crashes occurred. This did not work, and on top of that, it introduced a strong bias in the predictions (see Figure 13), if all fractions that one feeds the NN for training contain crashes, one filters out a significant part of the good beams. In this way, the NN does not see many successful beams and thus predicts lower beam currents.

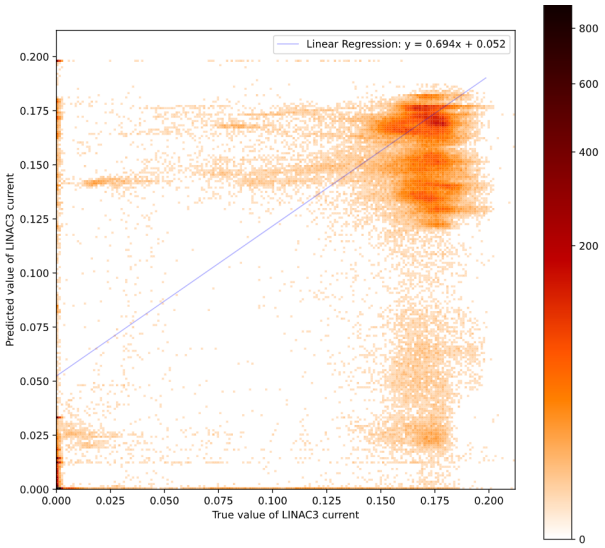


Figure 13: 2D-histogram of predicted vs true value of the LINAC3 current for the case where we skewed the data to contain more decays[2].

After actually plotting the predictions of the NN, and for the highest performing generic approach (gradient boosting random forests), we concluded that there is not enough information in a single time frame (even with rolling sum/variance applied) to predict the future beam current.

4.3 XGB model

For this approach, we use the exponential rolling averages from Section 2.1. Firstly, we analysed what time window (10, 20 or 30 minutes) was most promising for XGBoost. The time window of 30 minutes proved most suitable (it had the highest R^2, see Figure 14), so we chose to use the exponential rolling average with a time window of 30 minutes.

[2]Note that in these graphs it would be best to have all data points around the line $y = x$, since that would correspond with a perfect prediction.

In this dataset, we have not 14, but 42 features, since for each feature, we now also have a rolling mean and a rolling variance.

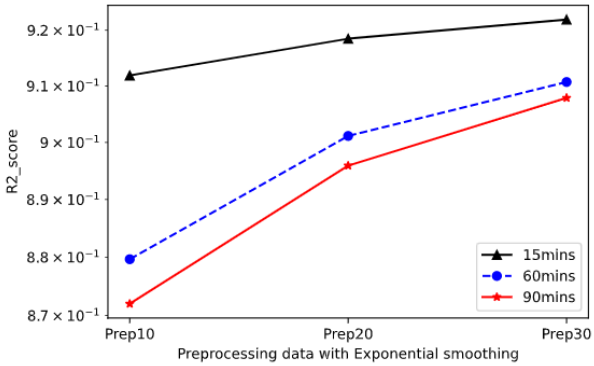


Figure 14: Using data preprocessed by three different windows to predict 15, 60 and 90 minutes ahead of "`ITL.BCT05:CURRENT`" target.

Second, we need to select features for our model. The variables consists three parts: the time, 14 number of setting variables that can be adjusted by operators, and 16 number of sensors variables that are considered a read only.
We mainly analyzed the following combinations of features:

- The oiginal dataset
- The rolling means
- The rolling means and rolling variances
- The original dataset, rolling means and rolling variances.

We did this for the 16 sensor variables without the 14 setting variables, and also for the 16 sensor variables together with the 14 setting variables.

By comparison, we discovered that the XGBoost model with only the rolling means of the sensor features achieved the best prediction result (the highest R^2 score and the lowest RMSE score) and so, we selected the rolling means of all sensor features, except for the feature `IP.NSRCGEN:RFTHOMSONAQNREV`, as it only contained zeros (see Appendix B). Since using 16 variables can result in computational difficulties, we chose the use PCA to reduce the dimensions, which allows us to reduce feature variables from 16 to 7.
For implementing our XGB model, we used a training dataset and a testing dataset. The training dataset corresponds with data from mid April until mid November 2018. As testing dataset, we used data from mid November until mid December 2018.

Now, it was time to test our approach and improve it by parameter tuning, some general approaches we tried are:

1. Choosing a higher learning rate. In general, the value of the learning rate is 0.1. However, for different problems, the ideal learning rate sometimes fluctuates between 0.05 and 0.3, and choosing the ideal number of decision trees corresponding to this learning rate. XGBoost has a very useful function called *cv*. It uses cross-validation in each iteration and returns the ideal number of decision trees to use in the XGB model.

2. For a given learning rate and number of decision trees, we can tune the following parameters of the decision tree by order to find optimal values for them. Those are the maximum tree depth, the minimum child weight, the gamma, subsample and lastly colsample_bytree).

3. Decrease the learning rate a bit and determine the ideal parameters again.

As a result, we have found that the following parameters for the linear XGBoost-Regressor worked best: colsample_bytree= 0.9, subsample= 0.9, learning_rate= 0.05, max_depth= 9, min_child_weight= 1, n_estimators= 3000, gamma= 0, alpha= 0.

To finally benchmark this approach, we used to our test dataset to validate the accuracy of the model. Here we divided the test data into a fitting and a forecasting part, in a 70/30 ratio, using random sampling. From Figure 15 above, the model seems very promising when applied to the testing dataset on random shuffle test data with a 30% test ratio. However, we should mainly focus on those regions of interest when the beam suddenly decays, as depicted in Figure 16.

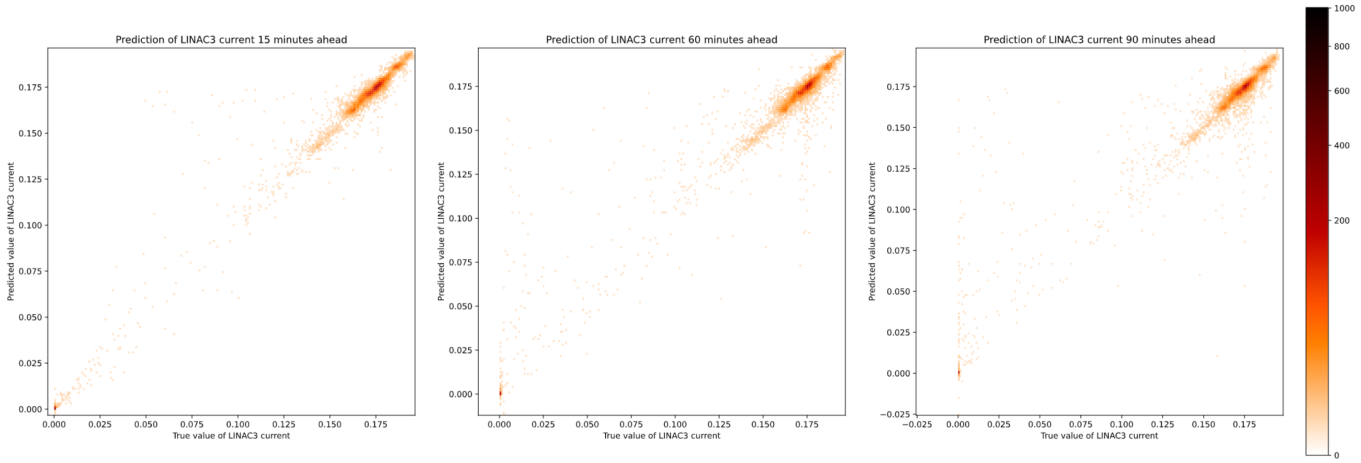


Figure 15: Results for the XGBoost for 15, 60 and 90 minute forecast with the 6 column set; Left (R^2: 0.9845, RMSE: 0.004988); Middle (R^2: 0.9588, RMSE: 0.007976); Right (R^2: 0.9458, RMSE: 0.009329)

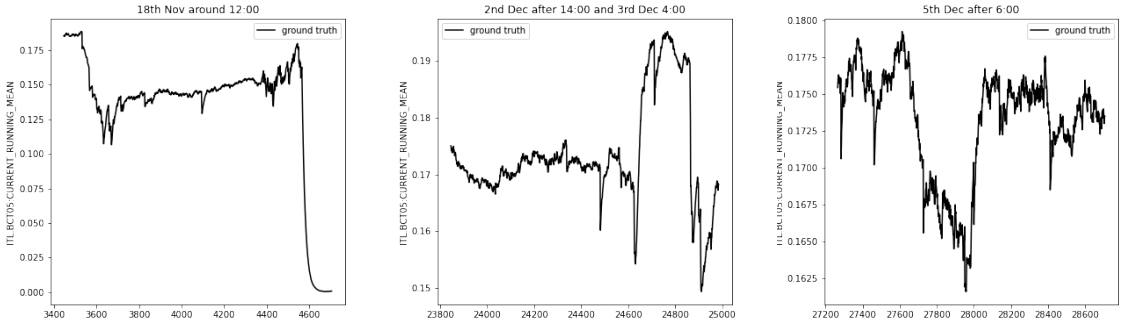


Figure 16: Different beam decays. Left: Start from Nov 18th at 11:00 until Nov 19th at 07:00; Middle: Start from Dec 2nd at 15:00 until Dec 3rd at 09:00; Right: Dec 5th the whole day.

Then, using the XGBoost model with the well-tuned parameters as mentioned above, we tried to forecast these three regions. However, different from before, we dropped one specific region data each time (such as the drop in the first specific region data, which starts from November 18th 11:00 to November 19th 07:00) and then used the rest of the testing dataset as the fitting part. The specific region data which was just dropped from the testing dataset is used as forecasting part. Repeating the aforementioned procedure three times, we then got our predictions for these three important regions. The results were the following:

As we can see, the predictions generally capture the flow of the ground truth values. It reflects the change tend well, but the variance of predictions is large and the fluctuations are pretty severe.
There is an exciting discovery: although we fail to predict the actual ground truth values (not even close), we can better capture the change trend of beam current which provides some useful information for operators. Since our main goal is to inform operators earlier before the beam current goes down and make the current as robust as possible, XGBoost model can achieve something closer to this goal.

Since we got seven features after using the PCA method, we can plot the feature importance graph based on our model and training dataset. There is an observation: Although the score value (F score) for each feature changes with different training dataset, it seems that the rank always remains the same. The rank is as follows: Feature0 $\rightarrow$ Feature1 $\rightarrow$ Feature3 $\rightarrow$ Feature2 $\rightarrow$ Feature6 $\rightarrow$ Feature4 $\rightarrow$ Feature5. The feature importance is calculated by the weight, representing is the number of times a feature appears in a tree (see Table 2).

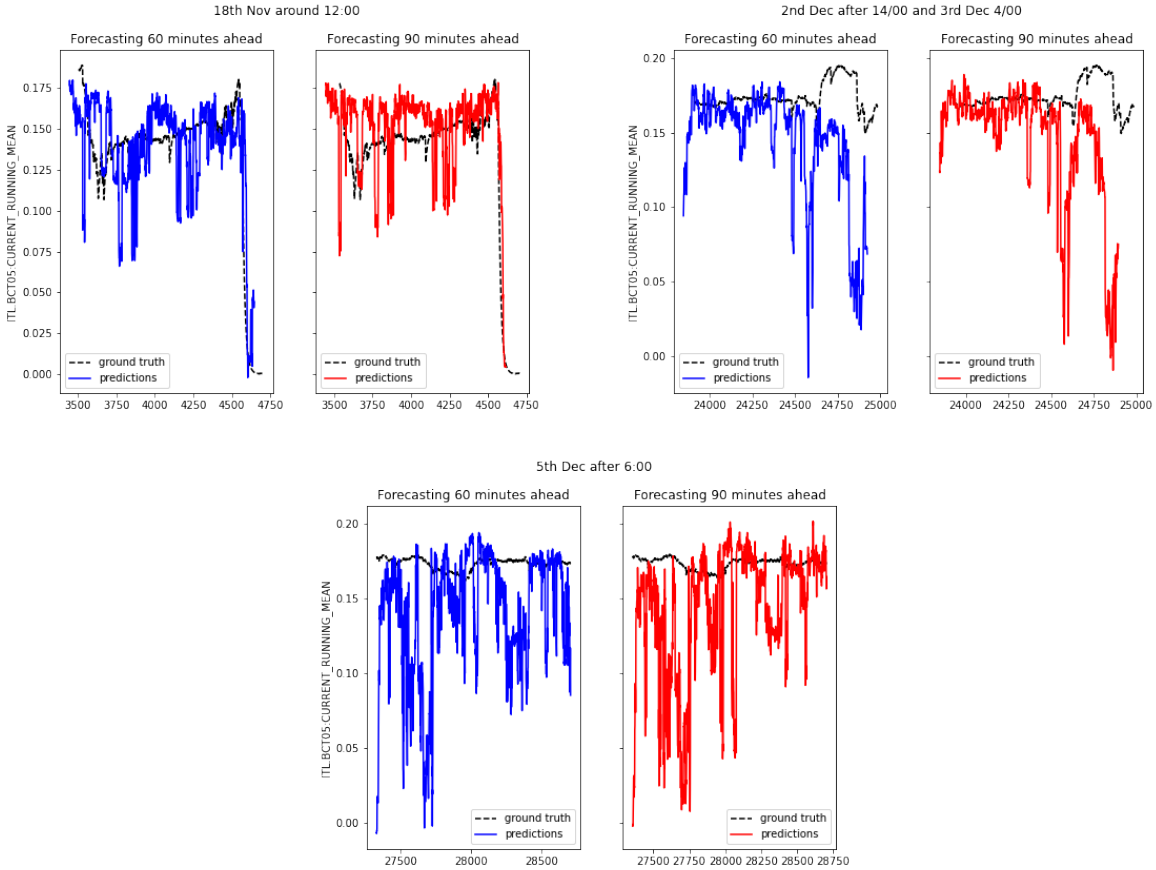


Figure 17: Predictions and ground truth values comparison.

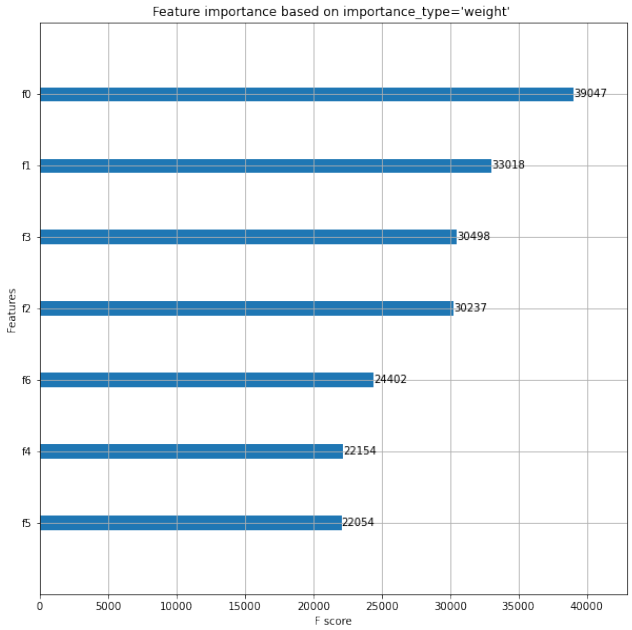


Figure 18: The data we use to train our model is test_df_30 drop out November 18th.

Feature	bias disk (mA)	GAS2 (P)	GAS3 (P)	Extraction (P)	sample lock (P)	Source injection (P)	oven1 (A)	oven1 (R)	oven1 (V)	Sairem	main source high voltage (mA)	intermediate electrode (mA)	Sairem2	oven2 (A)	oven2 (R)	oven2 (V)
0	0.36	-0.20	-0.07	0.23	0.15	0.30	-0.36	-0.13	-0.34	0.13	-0.37	0.36	-0.23	-0.17	-0.02	-0.18
1	0.18	-0.23	0.13	-0.06	0.22	0.04	0.18	0.25	0.23	0.03	-0.15	0.17	-0.33	0.40	0.46	0.40
2	0.04	-0.42	-0.19	-0.49	0.18	-0.39	-0.16	0.37	-0.18	-0.14	0.01	-0.01	0.13	-0.22	0.15	-0.22
3	-0.12	0.11	0.75	0.14	0.36	0.18	-0.07	0.27	-0.07	-0.04	0.08	-0.08	0.17	-0.20	0.10	-0.19
4	0.01	0.02	0.21	-0.13	-0.37	-0.19	-0.06	0.10	-0.03	0.85	-0.04	-0.01	-0.05	-0.05	0.03	-0.03
5	0.21	0.06	0.42	-0.12	-0.61	-0.19	-0.09	0.08	-0.05	-0.48	-0.23	0.07	-0.23	-0.02	-0.09	0.01
6	0.07	-0.20	0.37	-0.38	0.32	-0.22	0.06	-0.65	0.06	0.10	-0.04	-0.04	-0.09	0.05	-0.26	0.06

Table 2: Feature weights from the XGBoost approach with P: Pressure; mA, A: Current; V: Voltage; R: Resistance; Sairem, Sairem2: reflected power in W.

We also used the model for producing 3000 trees. For this, we actually used the testing data as our training data, but dropped out the decaying beam from November 18th to 19th (to be precise, from Nov 18th 11:00 am to 19th 7:00am) and as test data, we take this drop out part. We use the same parameters for our model as earlier. We visualised the first tree in Figure 19, the second tree is visible in Figure 20 in Appendix A.

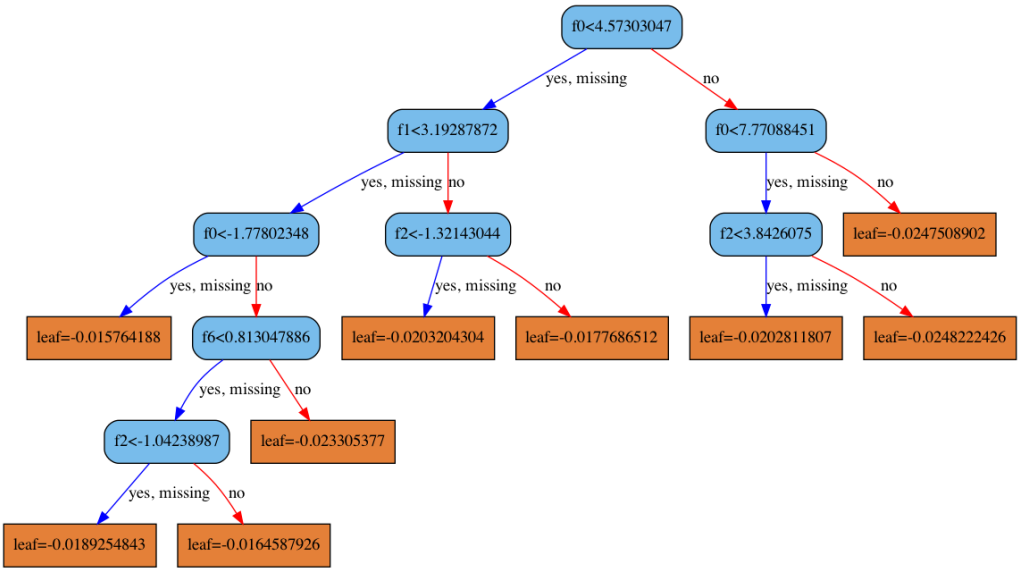


Figure 19: The first tree structure.

5 Conclusion

5.1 Summary

The main takeaway from our investigations is that *it is either not possible, or extremely difficult, to predict future beam decay based solely on observation of current beam state.* This conclusion was robust across models; even after applying sophisticated approaches based on current machine learning techniques, the ability of our models to forecast beam decay remained roughly on par with the persistence model.
Possible reasons for this include the following.

1. The fact that the operator is operating the machine could be removing patterns in the data; i.e. any "predictable" decay is already corrected for.

2. Beam decay could be *truly chaotic*, i.e. actually impossible to predict on sufficiently large time scales. This would comport well with the result that beam length is exponentially distributed, which implies that the beam

is Markovian.

3. Beam decay could be *highly path-dependent*, and only successfully predicted through an analysis of *dynamics* of sensor measurement that intelligently aggregated data from multiple times.

4. Beam decay could rely on *hidden internal state variables* that are not infer-able from current measurements.

However, although we were not able to produce a predictive model for beam decay, we have produced new insight into the beam decay problem that is potentially useful. From the feature selection and Granger causality research, we have identified subsets of columns that are worth looking into for future research, and additionally potential causations within the non-beam current columns. And finally, from the Ising model, we have given a model that flexibly segments the data.

5.2 Future Research

For each of the possible shortcomings of our models based on the current beam state, there are potential remedies.

1. If the inability to predict is coming from operators correcting for expected beam decay, then we could reduce our dataset to times where the operator is not on duty.

2. If the beam decay is truly chaotic, then we could try predicting on shorter time scales and then applying an automatic control scheme to keep the beam stable on short time scales.

3. If we need to look at the dynamics to predict the future... then we could look at the dynamics. This could involve manual feature augmentation to make the "time since last spike" available to whatever supervised learning algorithm we use, or other manual feature augmentation using "hidden state" that is infer-able from large-scale analysis of the data, but not from small-scale analysis.

4. In this case, we could politely ask the lab technicians to add more instrumentation to the beam apparatus.

Of course, it could be that all four of these are needed in order to automate beam current management.
More future research could involve taking the binary classifications from the Ising model and then trying to train a machine learning classifier, as opposed to a machine learning regressor. This classification could also be tweaked by modifying the Hamiltonian to be more sensitive to future values, so that we diagnose beam decay sooner. Another option we want to propose for future research as well is an approach for non-linear dimensionality reduction, namely manifold learning[17]. We ourselves were made aware of this too late into the project, however did not want to leave this method unmentioned.
Automating detection and recovery of beam decay is a very hard task, and we have only scratched the surface in this report. In this way, we were also not able to get our hands on the third objective of decision support. We look forward to seeing when the CERN team tackle this problem, and we hope that this report has moved them closer to that day.

5.3 Acknowledgements

Foremost, we want to sincerely thank Marc Bengulescu and Filip Široký from CERN for their guidance during this challenging project. We always could chat or call them when we had any questions. This allowed us to work more efficiently on the different methods we wanted to implement.
We thank CERN, Fieke Dekkers and Ivan Kryven for setting up this course for us.

References

[1] Alex Kohls, Jens Vigen, and Micha Moskovic. "Hadron Colliders, 50 years of discovery". In: *CERN Courier* 61.2 (2021), p. 39.

[2] Xingquan Zhu and Xindong Wu. "Class Noise vs. Attribute Noise: A Quantitative Study". In: *Artificial intelligence review* 22 (2004), pp. 177–210. URL: https://link-springer-com.proxy.library.uu.nl/article/10.1007%2Fs10462-004-0751-8.

[3] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. "Neural Ordinary Differential Equations". In: *arXiv:1806.07366 [cs, stat]* (Dec. 13, 2019). arXiv: 1806.07366. URL: http://arxiv.org/abs/1806.07366 (visited on 02/12/2021).

[4] David F. Nettleton, Albert Orriols-Puig, and Albert Fornells. "A study of the effect of different types of noise on the precision of supervised learning techniques". In: *Artificial Intelligence Review* 33.4 (Apr. 2010), pp. 275–306. ISSN: 0269-2821, 1573-7462. DOI: 10.1007/s10462-010-9156-z. URL: http://link.springer.com/10.1007/s10462-010-9156-z (visited on 03/23/2021).

[5] Robert Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996). Publisher: [Royal Statistical Society, Wiley], pp. 267–288. ISSN: 0035-9246. URL: http://www.jstor.org/stable/2346178 (visited on 04/14/2021).

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[7] Isabelle Guyon and André Elisseeff. "An introduction to variable and feature selection". In: *The Journal of Machine Learning Research* 3 (null Mar. 1, 2003), pp. 1157–1182. ISSN: 1532-4435.

[8] Tin Kam Ho. "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Proceedings of 3rd International Conference on Document Analysis and Recognition. Vol. 1. Aug. 1995, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994.

[9] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. "Regularization Paths for Generalized Linear Models via Coordinate Descent". In: *Journal of Statistical Software* 33.1 (2010), pp. 1–22. ISSN: 1548-7660.

[10] J. Friedman. "Stochastic gradient boosting". In: (2002). DOI: 10.1016/S0167-9473(01)00065-2.

[11] Sacha Friedli. *Statistical mechanics of lattice systems : a concrete mathematical introduction*. Red. by Yvan Velenik. New York, NY: Cambridge University Press, 2018. xix, 622 pages. ISBN: 978-1-107-18482-4.

[12] Richard McElreath. *Statistical Rethinking: A Bayesian Course with Examples in R and STAN*. Google-Books-ID: 6H_WDwAAQBAJ. CRC Press, Mar. 13, 2020. 575 pp. ISBN: 978-0-429-63914-2.

[13] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. 3rd ed. 2016. Springer Texts in Statistics. Cham: Springer International Publishing : Imprint: Springer, 2016. 1 p. ISBN: 978-3-319-29854-2. DOI: 10.1007/978-3-319-29854-2.

[14] C. W. J. Granger. "Investigating Causal Relations by Econometric Models and Cross-spectral Methods". In: *Econometrica* 37.3 (1969). Publisher: [Wiley, Econometric Society], pp. 424–438. ISSN: 0012-9682. DOI: 10.2307/1912791. URL: http://www.jstor.org/stable/1912791 (visited on 04/14/2021).

[15] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J. Király. "sktime: A Unified Interface for Machine Learning with Time Series". In: *arXiv:1909.07872 [cs, stat]* (Sept. 17, 2019). version: 1. arXiv: 1909.07872. URL: http://arxiv.org/abs/1909.07872 (visited on 04/14/2021).

[16] François Chollet et al. *Keras*. 2015. URL: https://keras.io.

[17] Ji Ham, Daniel Lee, and Lawrence Saul. "Learning High Dimensional Correspondences from Low Dimensional Manifolds". In: *Departmental Papers (ESE)* (Aug. 21, 2003).

A XGBoost Tree

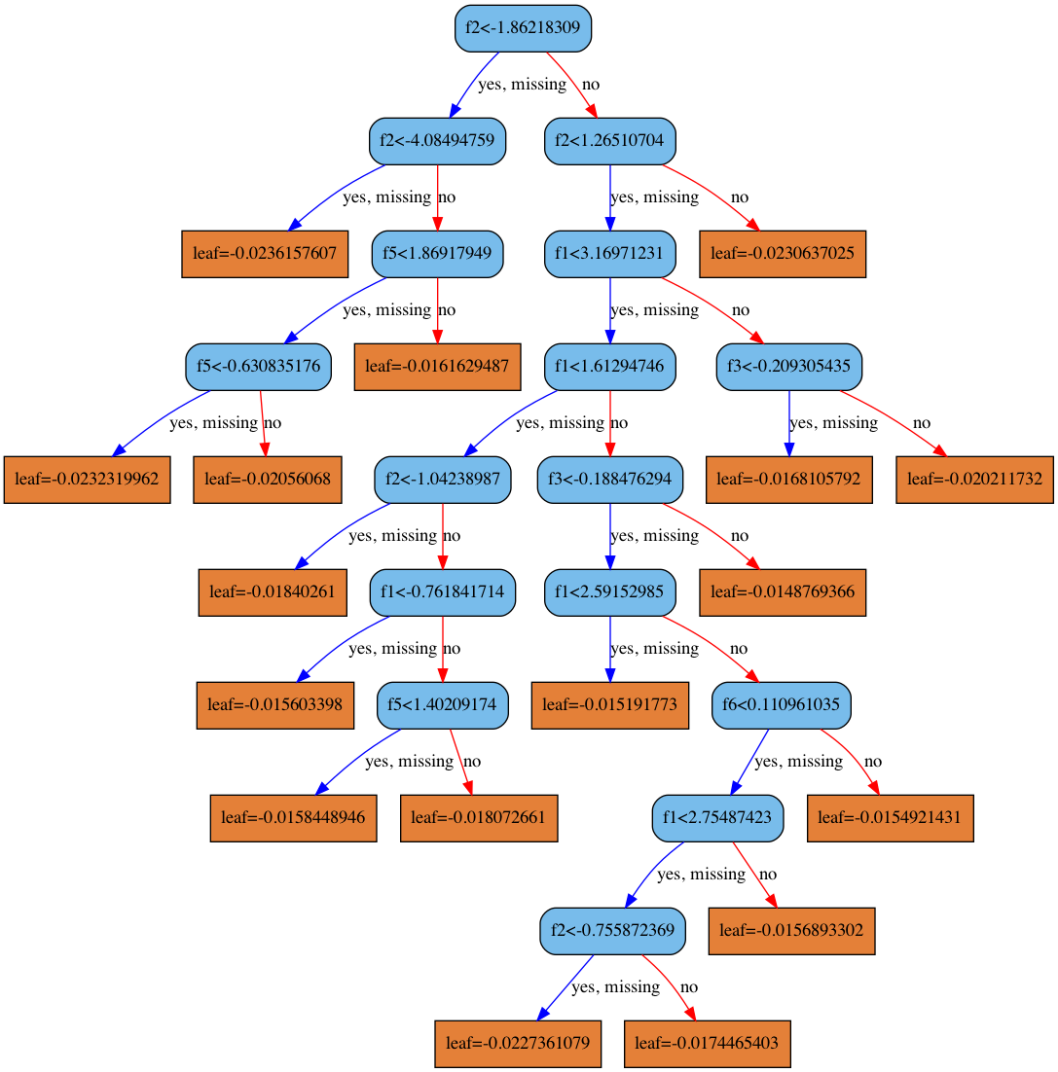


Figure 20: The second tree structure

B Data Columns

These are all the columns in the dataset we had received. Of these, CERN had proposed 7 important columns, which are highlighted in bold.

- "`Time`" Date time

Target features:

- **"`ITL.BCT05:CURRENT`" ion current of the outgoing beam in transformer `ITL.BCT05` in mA**
- "`ITF.BCT15:CURRENT`"
- "`ITF.BCT25:CURRENT`"
- "`ITH.BCT41:CURRENT`"

LINAC3 settings:

- "`IP.SAIREM2:FORWARDPOWER`" **forwarded power of the Sairem microwave generator in W**
- "`IP.NSRCGEN:RFTHOMSONAQNFWD`" forwarded power of the Thomson microwave generator in W
- "`IP.NSRCGEN:RFSAIREMAQNFWD`" forwarded power of the Sairem1 microwave generator in W
- "`IP.NSRCGEN:BIASDISCAQNV`" **voltage of the bias disk in V**

- "IP.SOLCEN.ACQUISITION:CURRENT" current of the source central solenoid in A
- "IP.SOLEXT.ACQUISITION:CURRENT" **current of the source extraction solenoid in A**
- "IP.SOLINJ.ACQUISITION:CURRENT" current of the source injection solenoid in A
- "IP.NSRCGEN:GASAQN" **voltage of the gas feedback loop in V**
- "IP.NSRCGEN:GASLOOP" number of the gas loop used
- "IP.NSRCGEN:OVEN1AQNP" **power of oven 1 in W**
- "IP.NSRCGEN:OVEN2AQNP" **power of oven 2 in W**
- "IP.NSRCGEN:SOURCEHTAQNV" voltage of the main source high voltage in V
- "IP.NSRCGEN:SOURCEINTERMEDIATEHTAQNV" voltage of intermediate electrode in V

LINAC3 sensors:

- "IP.NSRCGEN:RFSAIREMAQNREV" reflected power of the Sairem microwave generator in W
- "IP.NSRCGEN:RFTHOMSONAQNREV" reflected power of the Thomson microwave generator in W (note that this data column only contained zeros)
- "IP.SAIREM2:REFLECTEDPOWER" reflected power of the Sairem1 microwave generator in W
- "IP.NSRCGEN:BIASDISCAQNI" current of the bias disc in mA
- "IP.NSRCGEN:GAS2AQN" pressure at the GAS2 gauge in V
- "IP.NSRCGEN:GAS3AQN" pressure at the GAS3 gauge in V
- "IP.NSRCGEN:GASEXTRACTIONAQN" pressure in the extraction in V
- "IP.NSRCGEN:GASSASAQN" pressure in the sample lock in V
- "IP.NSRCGEN:GASSOURCEAQN" pressure in the source injection in V
- "IP.NSRCGEN:OVEN1AQNI" current of oven1 in A
- "IP.NSRCGEN:OVEN1AQNR" resistance of oven1 in Ohm
- "IP.NSRCGEN:OVEN1AQNV" voltage of oven1 in V
- "IP.NSRCGEN:OVEN2AQNI" current of oven2 in A
- "IP.NSRCGEN:OVEN2AQNR" resistance of oven2 in Ohm
- "IP.NSRCGEN:OVEN2AQNV" voltage of oven2 in V
- "IP.NSRCGEN:SOURCEHTAQNI" **current of the main source high voltage in mA**
- "IP.NSRCGEN:SOURCEINTERMEDIATEHTAQNI" current of the intermediate electrode in mA