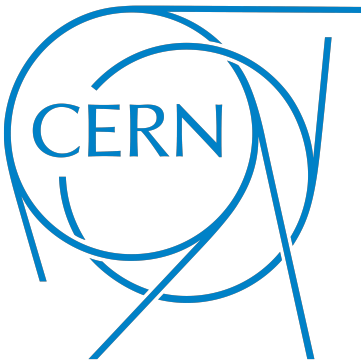
Master's Thesis Report

Predicting the ion beam current instability of LINAC3

Mathematical Institute
Utrecht University

Author: Manousos Emmanouil Theodosiou
Supervisor: Dr. Sjoerd Dirksen
Supervisor at CERN: Dr. Detlef Küchler
Second reader: Dr. Palina Salanevich

July 8th, 2022



Utrecht University

Submitted in partial fulfillment of the requirements for the degree of M. Sc.

Abstract

The GTS-LHC ion source provides heavy ions to the Large Hadron Collider (LHC) ion injector chain situated at the European Organisation for Nuclear Research (CERN) to conduct numerous physics experiments. The execution of such experiments relies on the stable operation of the ion source, which depends on frequent changes of the source's settings by a specialist. The objective of this research is to identify patterns in time series data from 2021 and forecast when the machine is going to fail, allowing the source specialist to take preventive action. In this study, a classical machine learning algorithm and five different neural network architectures were analysed and implemented to predict a beam decay. The results of the forecasting methods were compared to a baseline model to decide whether patterns of a beam decay exist. The implemented models were able to reach the performance of the baseline model but not surpass it. Given the current measurements, it is not possible to predict a beam decay in a short-term or long-term forecast. Additionally, two change point detection algorithms were provided to recognise abrupt changes in the status of the ion source on streaming data. After the implementation of a high voltage breakdown filter, it is possible to identify quickly and efficiently a beam decay in the present by reducing the number of false alarms.

Acknowledgements

I would like to thank Dr. Sjoerd Dirksen for his support and feedback during the thesis, Dr. Detlef Küchler for introducing me to this very interesting project and for his guidance throughout my internship at CERN and Dr. Palina Salanevich for being a part of this research. Also, I am grateful to my tutor Dr. Ivan Kryven for his advice and helping me find my path in Data Science. Last but not least, this work is dedicated to my family, friends and partner who were there for me in this journey.

Contents

1 Introduction

1.1 LINAC3

The GTS-LHC ion source at the linear accelerator 3 (Linac3) is an ion source where intense beams of heavy ions are produced [1]. Linac3 supplies accelerated ion beams out of lead and oxygen isotopes e.g. Pb^{54+} and O^{4+} respectivelly [1], to the Large Hadron Collider (LHC) ion injector chain [2]. The ion injector chain consists of Linac3, the Low Energy Ion Ring (LEIR), the Proton Synchrotron (PS), the Super Proton Synchrotron (SPS) and finally the Large Hadron Collider. It enables high energy collisions of heavy ion beams for the LHC experiments but also collisions on fixed targets for the SPS experiments. Due to their connection via the injector chain, the smooth operation of LHC relies heavily on the consistency of Linac3. A representation of the CERN accelerator complex can be found in Figure 1.

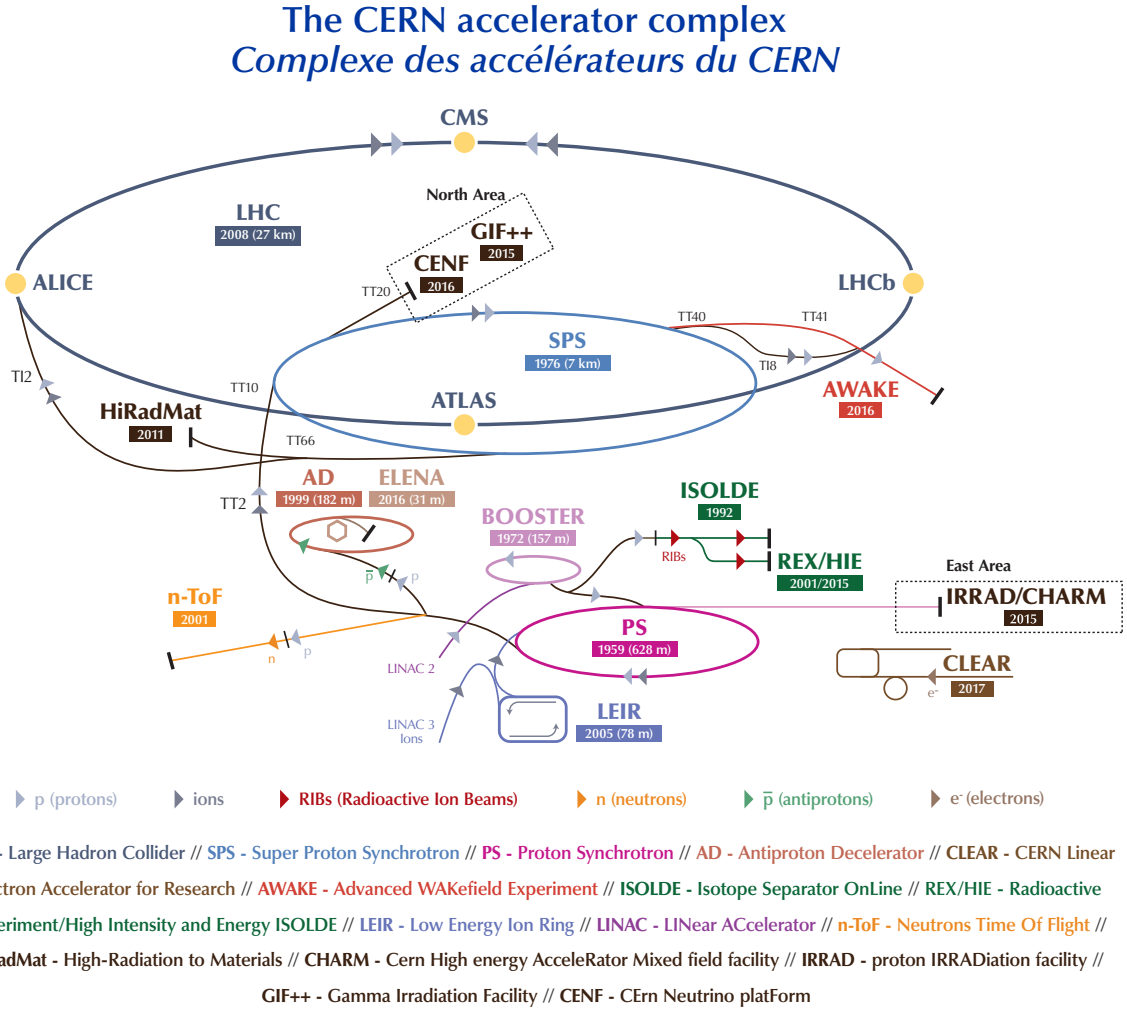


Figure 1: The Accelerator Complex at CERN [3]

Maintaining a good-quality beam is not a trivial task. Due to the multitude of physical phenomena that take place, Linac3 outputs a beam with varying intensity and large noise. The noise in the beam current can be interpreted as random flunctuations around its general trend. Changes in the trend of the beam intensity however can be thought of as beam variability. Due to changes in the source's status, the beam intensity decays and as a result it cannot be injected or used in other accelerators. An example of a beam decay can be seen in Figure 2. So far, maintaining the stability of the beam is a task that heavily relies on the experience of the operators (source specialists). Prior to our work, research has been done to implement machine learning methods to forecast the intensity of the beam. This will be discussed in Section 1.2. In Section 1.3 the structure of this research will be described.

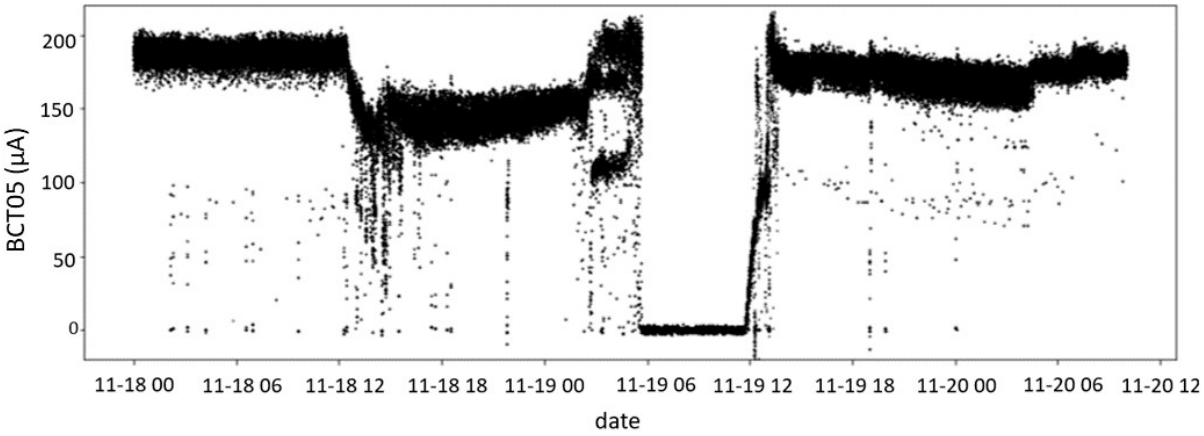


Figure 2: A beam decay is visible at 12pm on the 18th of November.

1.2 Previous research

Discovering patterns in beam decays is of great importance to ameliorate the stability of the GTS-LHC ion source. However, due to the complexity of GTS-LHC, it is not yet possible to determine the optimal settings for a stable beam. Previous research was focused on performing cluster analysis to identify sets of settings or patterns that could lead to a stable operation of the source without having the need of frequent interventions from the source specialists [4]. For the cluster analysis, the Optigrid algorithm was used on Linac3 data and the following conclusions were made [5]:

- Cluster analysis provided a broad amount of clusters that can not be merged into bigger ones without suffering the loss of important information
- There is no clear distinction between the settings that are accountable for a stable and unstable operation of the machine
- No clusters were found which provide settings leading to a stable beam

Prior analysis of Linac3 data has been carried out to predict the ion beam current intensity and more specifically the feature BCT05, which is the current of the ion beam after the GTS-LHC source [6]. In that analysis a plethora of machine learning models was implemented, although none of them was able to effectively forecast a beam decay before it occurs. The aim of this research was solely regression analysis [7] and the forecast horizons were 60 or 90 minutes into the future. As an example, Figure 3 is provided below depicting the predictions of the BCT05 rolling mean achieved by applying gradient tree boosting regression [8]. This model yielded the best predictions for the rolling mean of BCT05 compared to the other forecasting models which were implemented in that research.

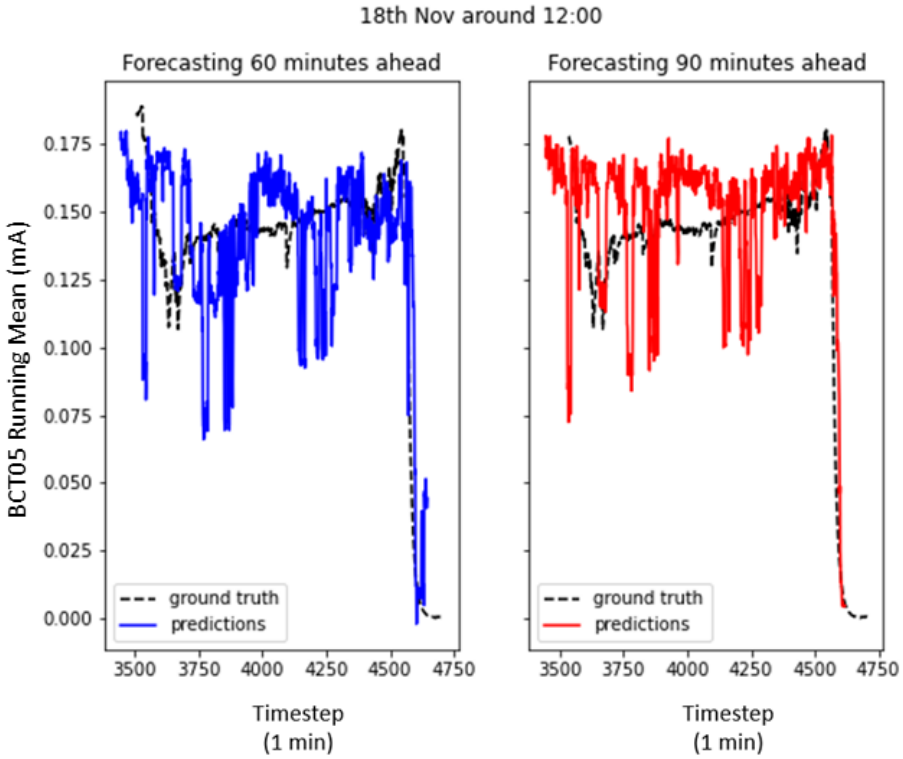


Figure 3: A comparison between the predicted values for the BCT05 current by implementing the XGBoost model against the rolling mean of BCT05 for a forecast horizon of 60 and 90 minutes respectively.

In Figure 3 the beam current predictions have high variability in comparison with the true values. What is more, the XGBoost model is not able to forecast a beam decay in advance.

1.3 Project description

Given the previous results, it is clear that predicting the ion beam current exactly with the currently available data might be extremely cumbersome or even impossible. A potential reason could be that the operators of Linac3 are frequently changing some settings of the machine to keep up the performance of source. This interferes with the predictive ability of any model. For instance, the model could predict a decay that never happened due to the fact that in the meantime the operators have changed the settings of the machine. These corrections cause the models to have high prediction error, urging the research to a different direction. Instead of predicting quantitatively the value of the beam current, its quality over time could be predicted as stable or unstable. This simplification can be done by transforming the regression problem into a binary classification problem as seen in Figure 4. Using the binary classification approach could potentially reduce the margin of error compared to regression since the target variable is binary instead of a continuous value.

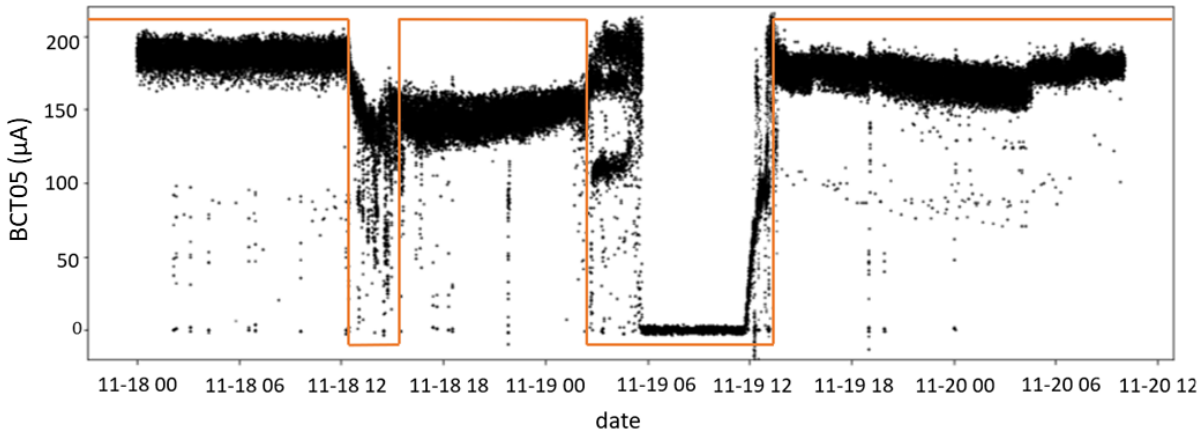


Figure 4: Example of classifying the beam as stable or unstable. With orange the stable periods are separated from the unstable periods. The decision for the above labelling is based on the changes of trend in the beam intensity that take place between midday and 4pm on November 18. What is more, high variability of the beam intensity is observed between 3am and 6am on November 19, outlining an unstable operation of the source. At 6am the source does not output a beam until midday. The ion source starts providing beams of stable intensity after a few minutes.

Taking this into consideration two questions can be distinguished. First, is beam instability occurring now? Change point detection algorithms [9] have shown great promise in identifying abrupt changes in time series. Given that the operator is not constantly monitoring the beam current, change point detection can be used as an alarm to take reactive action. Change point detection will be discussed more in Section 4.

Second, can a forecasting model predict whether a beam decay will take place in the future? If the beam indeed is about to decay it would be greatly significant to know when, in order to take proactive action. Although extensive research has already been done to forecast the Linac3 ion beam current in x minutes [6], the goal is to introduce classification models and compare them to a baseline model, all of them analysed in Section 3. Supervised learning methods and specifically classification can be implemented on time series to classify each data point as stable or unstable [10]. Training an accurate classification model could help forecast possible anomalies on streaming data and alert the operator. To prepare the data for binary classification, preprocessing is essential to train accurate and reproducible models. This will be explained in more detail in Section 2.

2 Data Description

For the purposes of this research, the data from Linac3 were analysed and preprocessed before they are fed into a model. The data are acquired by the Next CERN's Accelerator Logging Service (NXCALS) [11]. First, they are extracted using a Python wrapping of NXCALS, called pyTimber [12]. Moreover, data only from 2021 will be used, since the settings of the machine have been changing throughout the years. Therefore, making use of data from other years would not be representative of the current operation of the machine. In this chapter, the steps made to preprocess the data will be analysed. Next, two methods that can label the stability and instability of the beam current will be introduced, since there is no embedded feature from NXCALS that gives us this information.

2.1 LINAC3 data

The source specialists suggested to analyse the following features, which are linked to the settings and performance of the ion source.

Feature	Abbreviation	Value Range	Unit	Type
IP.NSRCGEN:BIASDISCAQNV	BIASDISC	$[-600, 0]$	V	Setting
IP.NSRCGEN:GASAQN	GASAQN	$[0, 10]$	V	Setting
IP.NSRCGEN:OVEN1AQNP	OVEN1	$[0, 20]$	W	Setting
IP.NSRCGEN:OVEN2AQNP	OVEN2	$[0, 20]$	W	Setting
IP.SAIREM2:FORWARDPOWER	FORWARDPOWER	$[0, 2500]$	W	Setting
IP.SOLINJ.ACQUISITION:CURRENT	SOLINJ	$[0, 1300]$	A	Setting
IP.SOLCEN.ACQUISITION:CURRENT	SOLCEN	$[0, 1300]$	A	Setting
IP.SOLEXT.ACQUISITION:CURRENT	SOLEXT	$[0, 1300]$	A	Setting
IP.NSRCGEN:SOURCEHTAQNV	HT_V	$[0, 22000]$	V	Setting
IP.NSRCGEN:SOURCEHTAQNI	HT_I	$[0, 5]$	mA	Acquisition
ITL.BCT05:CURRENT	BCT05	$[0, 0.3]$	mA	Acquisition
ITF.BCT25:CURRENT	BCT25	$[0, 0.1]$	mA	Acquisition
ITH.BCT41:CURRENT	BCT41	$[0, 0.1]$	mA	Acquisition

Table 1: A table containing all the features that will be used in this research.

Each feature is either a setting that can be changed and controlled directly or an acquisition, which in essence is a sensor that cannot be directly modified. More concretely, in Appendix A each feature is outlined. For completeness, the sketch of the main part of Linac3 is provided in Figure 5.

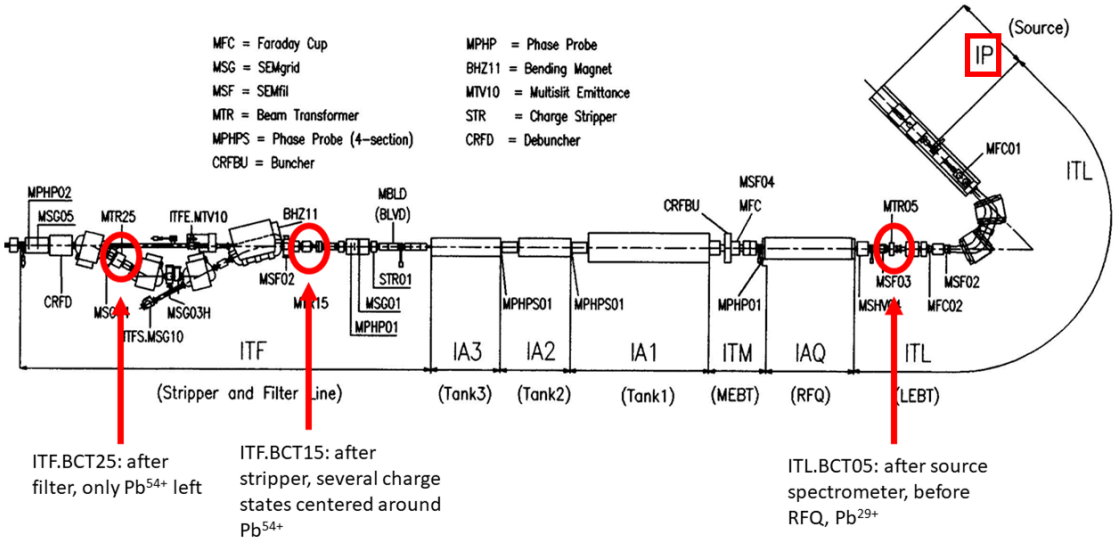


Figure 5: A blueprint depicting part of the machine inside the Linac3 hall. The BCT05, BCT15 and BCT25 (for full names of features see Table 1) are highlighted with a red circle. Most features for this research belong in the IP section, highlighted with a red square, where the ion source is situated.

2.2 Preprocessing

Primarily, each data point arrives in a non regular cycle, on average around 2 updates per second. A new datum, i.e. row element, gets stored in NXCALS as soon as there is an updated column feature. That is, a new setting or acquisition with its corresponding timestamp is saved only when its value has changed. Therefore, when the data is acquired from NXCALS and stored in a Pandas dataframe [13], each row has the value of the changed feature, the timestamp and all other features that do not change are stored as a NA value. For that reason, the first step is to forward fill the database's columns in order to take care of the NA values. The next step is to resample the data to 1.2 seconds, as each pulse of the beam is spaced by 1.2 seconds [14].

2.2.1 High Voltage breakdowns

Breakdowns of the source high voltage (HT) can take place. They do affect drastically the current with a sudden downward spike which lasts for a few seconds. After that, it is restored and goes back to its normal values without any external intervention. High voltage breakdowns are relevant to the performance of the ion source. However, due to the fact that the beam current value climbs back up, it cannot be considered as a beam instability. Breakdowns induce extra noise in the feature settings as depicted in Figure 7 and could therefore mislead any model that focuses on predicting beam instabilities.

Moreover, breakdowns are a frequent event that takes place in the machine. Another feature that has been added in 2017 on NXCALS, i.e. IP.NSRCGEN:SPARKS, measures precisely the number of HT breakdowns that arise on the source. On average, throughout 2021 approximately 406 breakdowns per month were recorded. This is also depicted in Figure 6. Thus, the goal is to filter these periods. As suggested in [4], the HT current has typically low variability. Hence, by implementing a threshold on this variability, i.e. sample variance to not exceed 0.25A, for a small rolling time window, i.e. 20 data points, the vast majority of high voltage breakdowns can be identified and filtered.

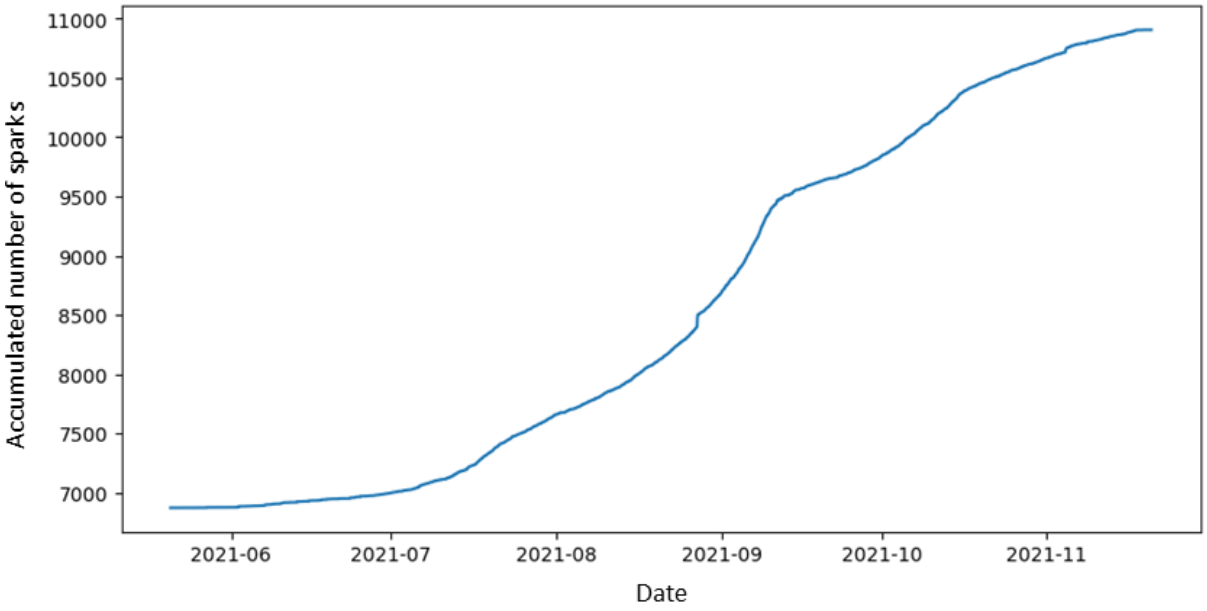


Figure 6: Accumulated number of HT breakdowns (sparks) over the last six months of operation in 2021.

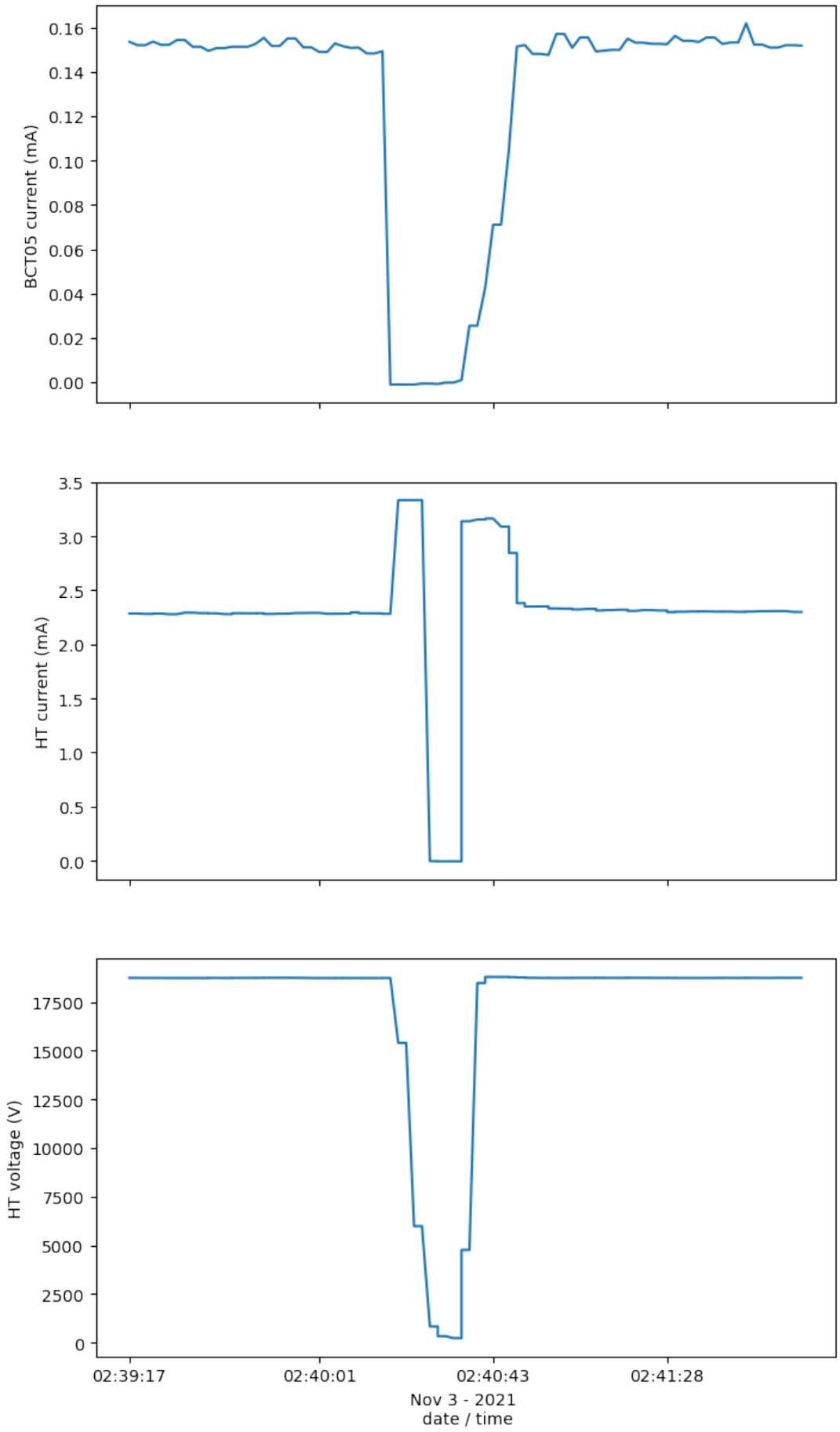


Figure 7: A high voltage breakdown takes place. The BCT05 current shows a sudden downward spike before it goes back to its normal values. The HT current soars shortly and plunges to 0A whilst the HT voltage drops to 0V.

2.2.2 Non desirable periods

Removal of outliers

Another step is to remove any feature readings that are deemed as unrealistic. For instance, negative oven power in Watts (W), or negative voltage of the gas feedback loop in Volts (V) are dropped. Also, a range of acceptable values for the features mentioned in Table 1 is given from the source specialists. Any measurement exceeding these limits is regarded as misleading and consequently the datapoint is omitted. Removing the above outliers improves the predictive ability of the models described in Section 3 since they do not learn on data with misleading measurements.

Oven refill

The process of creating lead ion beams starts by evaporating lead within the source plasma. Specifically, the oven is heating an amount of lead that needs to be replaced after around two weeks. After some significant modifications in the oven crucible [15], the number of oven refills was notably decreased. During an oven refill, as depicted in Figure 8, the oven stops and therefore no beam is produced. The times during an oven refill will not be taken into account for our analysis, as they are unrelated to beam instabilities. Therefore, these periods are omitted.

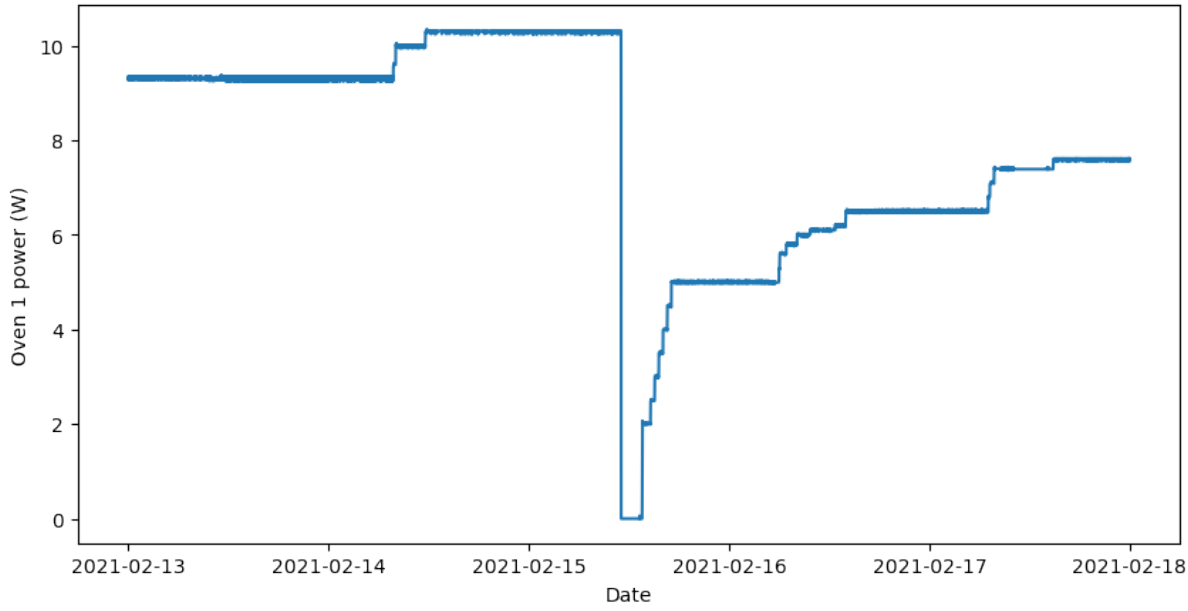


Figure 8: An oven refill takes place on the 15th of February around midday. The power of oven 1 goes to zero and after 3 hours the oven starts being recommissioned.

Downtime

There are also periods when no beam is measured from the transformer BCT05. Therefore, all the time intervals where no information about the performance of the beam is available, will not be taken into account. Specifically, any value below 0.01mA for the feature BCT05 will be filtered.

2.3 Labelling methods

As mentioned previously, there is no acquisition stored in NXCALS that could be used as a ground truth which classifies the beam as stable or unstable. Therefore, binary labels need to be created, i.e. 1 for stable and 0 for unstable, for each data point. It is critical that the labelling is accurate since the supervised learning method, i.e. classification model, will "learn" on these labels. Based on this training, the model will make predictions on unseen data and help the source specialists to make decisions whether an intervention on the machine is necessary. In the section below, two methods that were implemented to label each data point are proposed.

Rolling averages method

Taking into consideration previous work done at Linac3 [4], a time window can be classified according to its measure of spread, i.e. variance, and centre, i.e. mean. After various experiments a window of 1500 seconds was chosen to examine the mean of BCT25. In case the mean current of that window was ≤ 0.3mA, then all the data points belonging to that window are classified as unstable. Similarly, if the variance of BCT25 for this window was ≥ 0.01mA, then the data points in that window are classified as unstable. Finally, any data point with BCT25 ≤ 0.2mA was classified as unstable. The choice of these thresholds was made empirically after experiments on BCT25 data throughout 2021 and communicating the results to the source specialists. In principle, a correct labelling would require identifying the changes of trend in the BCT25 as instability whilst accepting low levels of noise as stable. An example of this labelling can be seen in Figure 9.

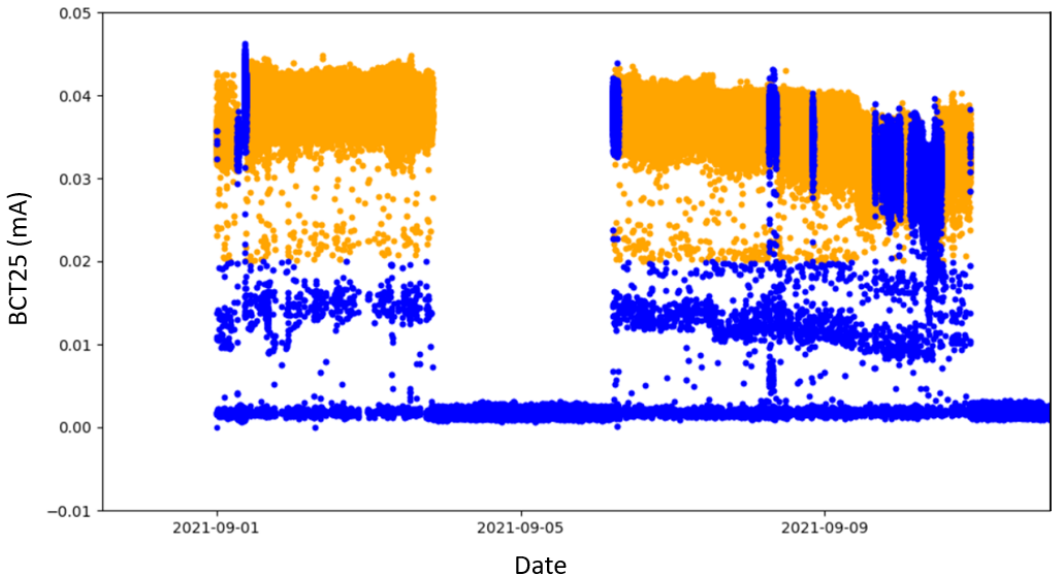


Figure 9: An example of labelling the data points as stable (orange) and unstable (blue).

Hand picked labels

Although the labelling in Figure 9 might be effective, there is still a considerable amount of data points that might be mislabelled. In order to bestow labels with higher accuracy, the time intervals of a decay can be identified using Timber. Timber is a CERN Graphical User Interface (GUI) application that provides interactive visualisations of time series data from NXCALS of any setting or acquisition in any given time and interval. Specifically, Timber was used in BCT05 data which measures the beam intensity right after the ion source. The criteria of this labelling relies on manually identifying changes of trend in the BCT05 data as in Figure 4 for small periods of time, i.e. one day, and noting the exact time when a beam decay started and ended. This process is repeated until the whole period of interest is checked. Finally, with the use of the `Pandas` library the time intervals noted as unstable are given the label 0 and label 1 otherwise. In this way, the margin of error for the labelling can be minimised, despite the fact that the process is not reproducible.

3 Forecasting methods

Time series classification (TSC) is a supervised learning task that has many different real-world applications in pattern recognition and anomaly detection, e.g. in medicine such as electrocardiogram (ECG) classification [16] or in cyber-security [17]. TSC is the task that predicts a target variable or a label that was assigned to time series data. The goal in the presented research was to predict a binary label

$$y_t = \begin{cases} 1 & \text{A stable beam at time } t \\ 0 & \text{An unstable beam at time } t \end{cases}$$

in forecast horizons of 15 or 60 minutes for given measurements $x_t \in \mathbb{R}^d$ of Linac3 at time t mentioned below.

Feature
BIASDISCAQNV
GASAQN
FORWARDPOWER
SOLINJ
SOLCEN
SOLEXT
SOURCEHTAQNI
BCT05

Table 2: A table containing all the features that will be used for the forecasts. The selection of these features is based on recommendations from the source specialists. All settings which belong in the source section are the most linked features to a beam decay, and BCT05 measures the beam intensity after the source.

Here $d = 8$ stands for the number of features provided in Table 2 that were used in this analysis. These features were selected since they are deemed by the source specialists as the most relevant to a beam decay. A successful model would be able to identify correctly patterns in the data that are associated with the unstable state of the beam. Moreover, the model can be used on unseen data, i.e. present data, to predict a beam decay that will take place in the future and inform the source specialists. Given these predictions, the source specialists can make decisions that could prevent a beam decay.

Deep learning has shown great promise in classifying time series effectively [18, 19]. Four out of five neural network architectures that are analysed in this section are motivated from [10, 20]. However, studies as in [21] also propose using classical Machine Learning (ML) methods such as Support Vector Machines (SVM) [22] instead of Artificial Neural Networks for time series forecasting. Therefore, in Section 3.2 an SVM model will be implemented to predict the stability/instability of the beam in the future given the other Linac3 measurements. Additionally, a new neural network architecture is proposed in Section 3.7. The predictive ability of the above models will be compared to a baseline model known as the persistence model. This model will be outlined in Section 3.1. Since a thorough analysis of machine learning (ML) and supervised learning in particular will not be provided here, the reader is referred to [23] for further background.

3.1 Persistence

There is plethora of machine learning algorithms to choose from and apply for time series forecasting problems. However, it is common practice to always implement a baseline prediction model beforehand in order to decide whether the more complex algorithms provide good forecasts. In this research the baseline model is called the persistence model.

The persistence model assumes that there has been no change in the stability/instability of the ion source between the present time and the future. More specifically, if at time t the beam is stable, i.e. $y_t = 1$, then the beam is forecasted to be stable for the next 15 or 60 minutes, i.e. the forecast horizon that is set. Particularly, $y_{t+1}, y_{t+2}, ..., y_{t+3000} = 1$ in case where $y_t = 1$. Here, the time interval between

two timesteps is 1.2 seconds as suggested in Section 2.2, and hence 3000 timesteps stand for a one hour interval. Similarly, the same rule can be followed for an unstable beam $y_t = 0$ at time t.

Once all the models mentioned in Section 3 are implemented, their results can be compared to the persistence model. If their predictive ability is below the baseline model's, then it can be concluded that it is not possible for them to effectively predict a beam decay.

3.2 Support vector machines

Support vector machines (SVMs), introduced in [24], have numerous implementations in both regression and classification problems. What is more, SVMs have been applied on forecasting financial time series and performed better than several ANN architectures in [21]. Therefore, it is desirable to apply SVMs in this research problem and compare the results to more complex models outlined later in this section.

In the classification setting, the algorithm is looking for a hyperplane $H(a, b) = a^\top x + b$, where $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$, that can separate two sets of points $x \in \mathbb{R}^n$ with different labels $y \in \{-1, +1\}$, in a way that their distance is maximal [1]. Hence, a classifier $f(x) = \text{sign}(a^\top x + b)$ can now be defined.

First, let's assume that the two sets of points x with label y are separable. Thus, the following rule can be applied to separate x_i given its label y_i for $i = 1, ..., t$

$$y_i(a^\top x_i + b) \geq 1 \quad i = 1, ..., t \tag{1}$$

note that $y_i(a^\top x_i + b) - 1 = 0$ for x_i called *support vectors* that lie exactly on the maximum-margin hyperplane. A relaxation of Equation (1) is possible with the introduction of slack variables $\xi_1, ..., \xi_t \geq 0$ in the case when x_i are not separable

$$y_i(a^\top x_i + b) \geq 1 - \xi_i \quad i = 1, ..., t \tag{2}$$

The maximum-margin hyperplane, the slack variables and the support vectors are depicted in Figure 10.

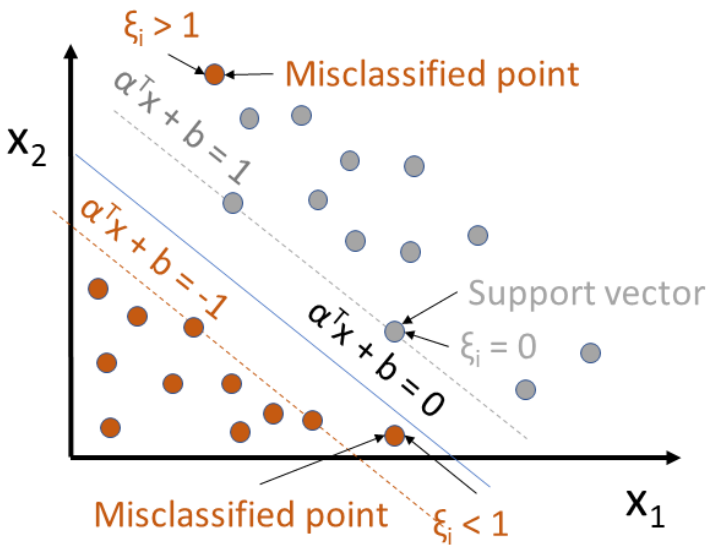


Figure 10: A 2-dimensional example of a maximum-margin hyperplane for an SVM trained with samples from two sets of points with different labels (denoted with orange and gray). Samples on the maximum-margin hyperplane are called support vectors. The slack variable ξ_i measures by how much the constraint in Equation (2) is violated.

From Figure 10 it can be concluded that any other observation apart from the support vectors and the misclassified points is irrelevant to the model's binary class boundaries. The proof of the above

[1] Note that the labelling $y \in \{-1, +1\}$ used in this section is more convenient for the mathematical formulation of SVMs compared to $y \in \{0, 1\}$. Algorithmically speaking, there is no difference between the two labellings. In this research the SVMs were trained using the labelling $y \in \{0, 1\}$.

statement, which is based on the *Karush-Kuhn-Tucker* conditions [25], can be found in the Appendix A. The aforementioned observation is a significant advantage of SVMs compared to other learning algorithms due to their robustness to outliers and noise from Linac3 time series data.

With at least two support vectors, $x_{(+)}$ with label $+1$ and $x_{(-)}$ with label -1 the width of the maximum-margin hyperplane is

$$\text{width} = \frac{2}{||a||} \tag{3}$$

The proof can be found in the Appendix A. The objective is to maximise the width of the maximum-margin hyperplane $\frac{2}{||a||}$ which is identical to minimising $\frac{||a||}{2}$. Additionaly, the sum of the slack variable ξ also needs to be minimised, as it is desirable to reduce the number of misclassifications. More concretely

$$\begin{aligned} \underset{(a,b,\xi)\in\mathbb{R}^n\times\mathbb{R}\times\mathbb{R}^t}{\text{minimize}} \quad & \frac{1}{2}||a||^2 + C1^\top\xi \\ \text{subject to} \quad & y_i(a^\top x_i + b) \geq 1 - \xi_i \quad i = 1, ..., t, \\ & \xi_i \geq 0 \qquad\qquad i = 1, ..., t \end{aligned} \tag{4}$$

where Equation (4) is called the *primal problem* and C is a constant factor with which every misclassification $\xi_i > 0$ is multiplied and later added to the objective function. Thanks to the concept of duality, optimisation problems can be perceived in two possible ways, the primal and the dual problem. As a result, the dual optimisation problem can be optimised instead of the primal. Additionally, the Lagrangian dual problem provides the best alternative especially in the case where strong duality is possible. That is,

$$p^\star = d^\star \tag{5}$$

where $p^\star, d^\star$ denote the optimal values for the primal and dual problem respectively. For the primal problem demonstrated in Equation (4), strong duality holds since the primal problem is convex and Slater's condition holds [25]. That is true since the objective function is convex and the inequality constraints are affine. Since strong duality holds, it is desirable to solve the dual optimisation problem instead of the primal. Explaining in detail convex optimisation is out of scope of this research. For further information about convex optimisation, duality, the Lagrange dual function and affine functions the following literature is recommended [25]. Finally, from [23, 26] the following dual optimisation problem for the SVM learning algorithm is derived

$$\begin{aligned} \underset{\lambda\in\mathbb{R}^t}{\text{maximize}} \quad & \sum_{i=1}^{t}\lambda_i - \frac{1}{2}\sum_{i,j=1}^{t} y_i y_j \lambda_i \lambda_j x_i^\top x_j \\ \text{subject to} \quad & 0 \leq \lambda_i \leq C \quad i = 1, ..., t, \\ & \sum_{i=1}^{t}\lambda_i y_i = 0 \end{aligned} \tag{6}$$

where λ_i for $i = 1, ..., t$ are called Lagrange multipliers. Notice that the formulation of SVMs thus far applies only to linearly separable classes as depicted in Figure 10. However, applying a kernel $K(x_i, x_j)$ instead of a dot product $x_i^\top x_j$ in Equation (6) would allow performing non-linear decision boundaries for classification instead of linear ones. This is also known as the kernel trick [23].

For the purposes of this research, different types of kernels have been tested on the SVM model to forecast the beam instability of Linac3. Apart from the linearly separable case, the following kernels will be used as these were proposed for time series forecasting problems in [21]:

- Polynomial kernel: $K(x, x') = (1 + x^\top x')^k$
- Radial basis function (RBF) kernel: $K(x, x') = \exp\left(-\frac{||x-x'||^2}{2\sigma^2}\right)$
- Sigmoid kernel: $K(x, x') = \tanh(cx^\top x' + d)$

where $x, x' \in \mathbb{R}^n$. In principle, SVMs are a quite versatile learning algorithm for classification problems that can be utilised to forecast a beam instability. Its robustness to outliers and noise on time series data is another factor that influenced the decision to choose this model. On the other hand, SVMs are not probabilistic classifiers, i.e. they do not yield a probability for the classifications they provide and the choice of kernel is a non-trivial task.

3.3 Multilayer perceptron

Deep learning has shown a great surge in popularity in the early 2010s and started to dominate in fields such as computer vision with AlexNet in 2012 [27]. Due to the gigantic increase in computational power in both CPUs, GPUs and the availability of vast amounts of data, the application of ANNs was made possible, although the idea of ANNs was introduced in the past. The perceptron, as formulated in [28], is the structural building block of neural networks. In essence a perceptron is a hyperplane $W^\top X + b$, with $W = (w_1, ..., w_m)^\top$ called weights and b the bias term, which classifies an input $X = (x_1, ..., x_m)^\top$ as 1 if $W^\top X + b > 0$ or 0 otherwise.

This idea was developed further by applying a non-linearity g, also known as activation function, having the above affine transformation of X as an input

$$h^{(1)} = g^{(1)}\left(\mathbf{W}^{(1)\top} X + b^{(1)}\right) \tag{7}$$

with $^{(1)}$ denoting the first layer of the neural network. The output of the non-linearity is also called a neuron. In neural networks, the number of neurons can be specified, i.e. the width, that will be put in parallel forming a layer. In a fully connected network, each neuron of a layer l_1 can be mapped to each neuron of the next layer l_2. First, the neuron in l_2 receives the sum of all its connections, where each connection has its own weight, then adds a bias term and finally activates the result by feeding it to a non-linearity

$$h^{(l_2)} = g^{(l_2)}\left(\mathbf{W}^{(l_2)\top} h^{(l_1)} + b^{(l_2)}\right) \tag{8}$$

Fully connected layers are also called dense. This process continues until the output layer is reached. Therefore, depending on the number of neurons per layer and the number of layers, neural networks can have thousands of parameters that need to be learned. Thus, it is computationally expensive to train neural networks and for them to learn the necessary parameters that allow them to generalise well on unseen data.

Non-linearities are used because it is very frequent that non-linear dependencies exist in data from real-world applications. In cases where there are linear trends in the data, neural networks should be avoided and instead a less complex learning algorithm such as SVMs should be implemented. Given the Universal approximation theorem [29], which states that neural networks with one hidden layer and sufficient amount of hidden units have the ability to estimate any Borel measurable function [30], neural networks were applied in this research to recognise the patterns of a beam decay and predict it.

Multilayer Perceptrons (MLPs) are deemed as the most classic type of feedforward artificial neural networks. In the Linac3 case, the network is fully connected and in each layer a non-linearity, i.e. the Rectified Linear Unit (ReLU) activation function, is applied. ReLU is a piecewise linear activation function defined as

$$\mathrm{ReLU}(z) = \max(0, z) \tag{9}$$

for $z \in \mathbb{R}$ being an input. It provides faster training since it can output a zero but also it is computationally trivial to apply. Another advantage of ReLU is that it diminishes the Vanishing Gradient Problem. In principle, the parameters of each neuron in the neural network W and b get optimised. This is done through gradient based methods [31] such as gradient descent to minimise a loss function. The parameters get updated through backpropagation [32]. During backpropagation, gradient information is passed from the output layer back to the input layer. However, the gradients that are updating the parameters of the neural network might get increasingly smaller. This causes the model's parameters to stop getting updated whilst the model is not yet optimised. Based on this observation, the input data coming from Linac3 are normalised.

Dropout is also used, which allows to reduce randomly the number of active neurons in a layer. This process decreases the probability of overfitting [33].

In the output layer the softmax activation function is applied to compute the probability of whether a Linac3 data point will be stable or unstable. The softmax activation function for binary classification is defined as

$$\sigma(z_i) = \frac{e^{z_i}}{e^{z_1} + e^{z_2}} \tag{10}$$

where $z_i \in \mathbb{R}$ is input for the softmax layer and $i = 1, 2$. By taking probabilistic decisions in a classification task, the certainty of a forecast can be quantified, as opposed to SVMs which by default do not provide a probability.

3.4 Convolutional neural networks

Convolutional neural networks (CNNs) are known for their capacity to effectively capture patterns in data, which have temporal or spatial dependencies [34]. Furthermore, recent research in time series classification problems using CNNs have shown promising results [18]. CNNs that exploit the temporal information of Linac3 data are applied in this research to forecast a beam instability.

Convolutional neural networks are comprised by convolutional layers, in which a kernel K of size k is applied. In the one-dimensional setting, a kernel $K \in \mathbb{R}^k$ from the convolutional layer is multiplied element-wise with a time series $x = (x_t)_{t=0}^{N-1}$ and the products are then subsequently added. This can be formulated in the following way

$$(K * x)(t) = \sum_{i=0}^{k-1} K(i) x_{t-i} \tag{11}$$

The $*$ operation is called convolution. The kernel slides through the time series data with a user-given stride. This operation is demonstrated in Figure 11.

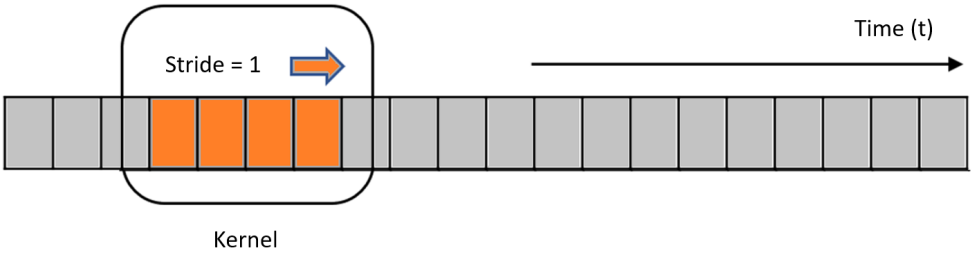


Figure 11: Example of a one-dimensional convolution on a time series. The one-dimensional kernel with size $k = 4$ moves along the time series with one timestep, i.e. stride is one.

After a convolutional layer, batch normalisation [35] is performed to reduce the number of required epochs to train the model as well as to avoid overfitting. Subsequently, a non-linearity is applied, i.e. ReLU, and the result is fed to the next convolutional layer and the same process is repeated. The above convolutional block can be formulated in the following way as in [10]

$$z = K * x + b \tag{12a}$$

$$n = \mathrm{BN}(z) \tag{12b}$$

$$h = \mathrm{ReLU}(n) \tag{12c}$$

where BN stands for batch normalisation, x denotes time series data from Linac3, K is the kernel, b is the bias term and $*$ denotes the convolution operation. It is important to note that in the course of training, as opposed to MLPs, the CNN optimises the weights of the kernel which are shared within a feature map. As a result, CNNs have significantly less parameters to optimise and they are less prone to overfitting.

Before the output layer, the global average pooling layer [36] is applied instead of a fully connected dense layer. Each feature map, i.e. the output of a kernel being convolved with the time series, gets averaged,

outputting only one value per feature map. The values from the global average pooling layer are then fully connected to a softmax layer outputting a probability as in the MLP model.

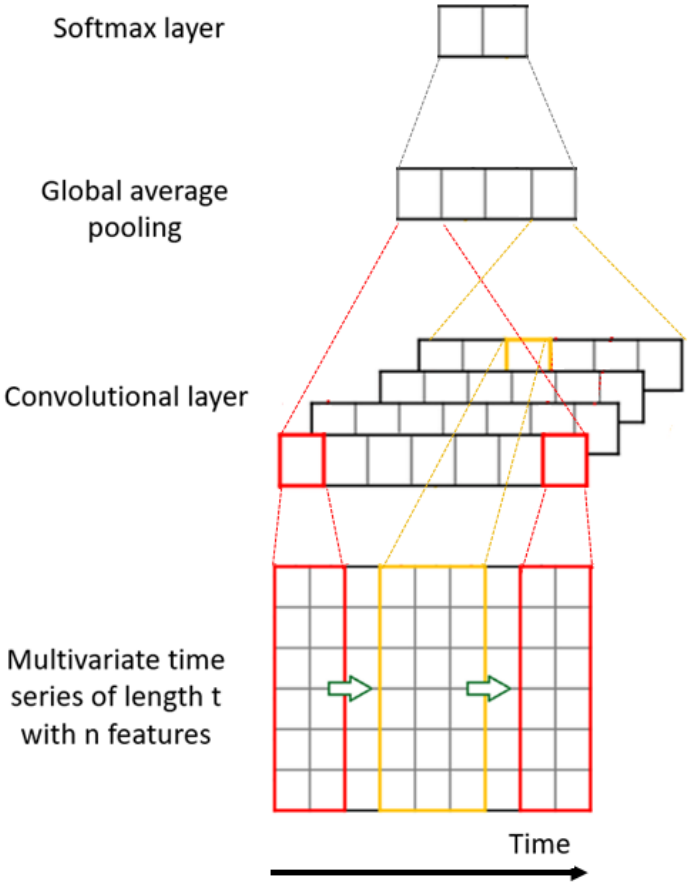


Figure 12: An example of four kernels being convolved with a n-dimensional time series. The results from the convolution get aggregated per feature map. The vector coming after the aggregation constitutes an input for the softmax layer, which produces the probability for a binary variable, e.g. stable or unstable beam. Figure motivated from [37].

In order to respect the ordering of the time series data and make a prediction more precise, causal padding is utilised. In causal convolutions the forecast made by the CNN for timestep t, does not "see" the observations that occur in both the present and the future, i.e. $x_t, ..., x_N$, where N is the length of the time series. This is possible, in the case of a convolutional layer with kernel size K, by padding the input with $K-1$ zeros at the front of sequence $[0, ..., 0, x_0, ..., x_N]$.

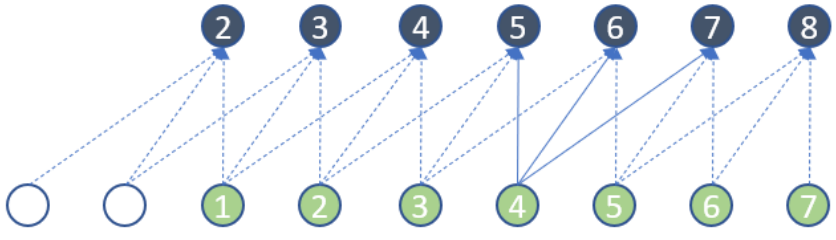


Figure 13: A CNN with kernel size $K = 3$ and stride of step one is used to predict the target variable (blue) at the next time step given the observations (green) using causal padding. Figure motivated from [38].

Although recurrent neural networks are said to be state-of-the-art models to forecast time series, CNNs can also provide a great alternative with less training time [39]. Since there are no recurrent connections [40] in CNNs, the receptive field, i.e. the set of datapoints in the input layer that can influence one node in the output layer of the CNN, will be expanded. For this reason, a deeper version of CNNs was implemented.

3.5 Dilated convolutional neural network

CNNs are highly capable of extracting features either in one-dimensional or two-dimensional data. For example, a convolution of a kernel $[\frac{1}{k}, ..., \frac{1}{k}]$ of size k and stride equal to one with a time series computes the moving average of that time series. A convolution could make noisy Linac3 data, such as the BCT05, smoother and keep the important features when, for example, BCT05 begins to decay. More importantly, it is desirable to identify long-term dependencies between the observations from Linac3 and the stability of the beam. For this to be possible, dilated convolutions are used.

Dilated convolutions allow the kernel to skip d nodes, where d denotes the dilation at a given layer $l = 1, ..., L$. Hence, the dilated convolution on a univariate time series $x = (x_t)_{t=0}^{N-1}$ can be formulated as in [41]

$$(K *_d x)(t) = \sum_{i=0}^{k-1} K(i) x_{t-d \cdot i} \tag{13}$$

Note that in dilated convolutions the kernel has a stride of one time step. Similar to Section 3.4 and Section 3.6, causal padding is used to let the model take into account only the past observations of Linac3.

Expanding the receptive field of an output node is possible by stacking layers of dilated convolutions. By increasing the dilation exponentially at each layer, i.e. $d \in [2^0, 2^1, ..., 2^{L-1}]$, the receptive field increases at the same rate [20]. This is highly desirable, since the growth of the receptive field is linear in CNNs, depends on the number of convolutional layers in the network and the size of the kernel [42]. It is important to highlight that in these dilated convolutional neural network models the stride was set to one time step, since it is disadvantageous to skip nodes at the input layer.

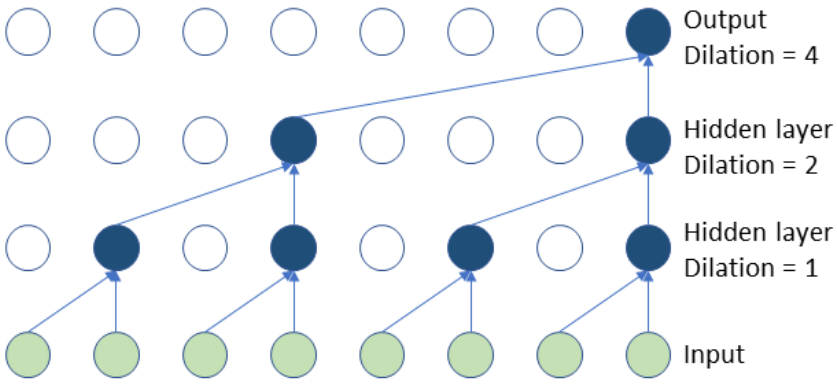


Figure 14: An example of a dilated convolutional neural network with kernel size 2. The receptive field is equal to 8. Figure motivated from [20].

Given a stack of dilated convolutions, the receptive field f can be calculated in the following way

$$f := 2^{L-1} k \tag{14}$$

where L denotes the number of stacked dilated convolutional layers and k the size of kernel K. To predict a beam instability, multiple layers of dilated convolutions were stacked to increase the receptive field followed by a ReLU non-linearity. Finally, a softmax layer in the output provides the probability to classify a stability/instability.

3.6 Residual network

Given that CNNs can be utilised to make effective predictions on time series data, a deeper neural network architecture is applied. The neural network proposed in this section is called the residual neural network (ResNet), first introduced in [43]. In these networks, a shortcut connection between the building blocks of the neural network, called residual blocks, is introduced to mitigate the vanishing gradient problem which is prominent in deep neural architectures. For a deep neural network with k layers, a small gradient will have to be multiplied k times to be backpropagated to the input layer. Therefore, the information back to the input decreases exponentially as the number of layers is increased. However, with the shortcut connections it is possible to skip a user-given number of layers.

For example, let an activation h^l of a node at layer l which gets multiplied by a weight and added by a bias term, i.e.

$$r^{l+1} = W^{l+1}h^l + b^{l+1} \tag{15}$$

Hence, the activation i.e. ReLU at layer $l+1$ is

$$h^{l+1} = \text{ReLU}(r^{l+1}) \tag{16}$$

Similarly for layer $l+2$

$$h^{l+2} = \text{ReLU}(r^{l+2}) \tag{17}$$

where $r^{l+2} = W^{l+2}h^{l+1} + b^{l+2}$. This is the normal path that a node follows from layer l to layer $l+2$. The shortcut allows the activation of the node at layer l to be added directly to layer $l+2$ along with the output of the normal path just before the activation function in layer $l+2$ is applied

$$\hat{h}^{l+2} = \text{ReLU}(r^{l+2} + h^l) \tag{18}$$

The main path from layer l to $l+2$ along with the shortcut connection constitute a residual block. In the presented research, the shortcut used is an identity function which does not make the model more complex than it already is, since no additional parameters are introduced. The shortcut avoids all the intermediate activation functions of the normal path, which could shrink its derivative. Thanks to this fact, the overall gradient of the block will be increased, enhancing the flow of information which updates the parameters of the neural network. In this way, more layers can be stacked compared to Section 3.4.

By managing to diminish the Vanishing/Exploding gradient problem, literature in [43, 18] suggests that ResNets are able to potentially minimise faster the loss function and therefore make the training more powerful than architectures with no shortcut connections. Moreover, by omitting redundant layers ResNets are allowed to train in shallower architectures when there is no improvement in accuracy. In this way, ResNets choose implicitly the number of layers that would bring added value to the model in a more dynamic way. Taking all this into consideration and given the performance shown by ResNets in time series classification problems in [18], a ResNet was implemented to forecast a beam decay.

In this research, a residual block is made of three convolutional layers, each followed by batch normalisation and a non-linearity, i.e. ReLU. This can be formulated in the following way as in [10]

$$a_1 = \text{ConvBlock}_{k_1}(x) \tag{19a}$$

$$a_2 = \text{ConvBlock}_{k_2}(a_1) \tag{19b}$$

$$a_3 = \text{ConvBlock}_{k_3}(a_2) \tag{19c}$$

$$s = a_3 + x \tag{19d}$$

$$\hat{h} = \text{ReLU}(s) \tag{19e}$$

where ConvBlock is a convolutional block as described in Equation (12), k_i is the kernel for each convolutional block $i = 1, 2, 3$, s is the residual block having a shortcut x and $\hat{h}$ is the activation of the residual block. The ResNet model that was implemented to forecast a beam instability consists of multiple residual blocks. The last residual block is then connected to the output layer in a similar way as the CNN described in Section 3.4.

In principle, by increasing the number of layers in a neural network, the complexity of the model is also increased. This could potentially allow the model to learn more accurately the training data. Consequently, deep neural networks have a high chance of overfitting due to the large amount of parameters that exist in the model and might not be able to generalise effectively. Furthermore, deep neural network architectures are more computationally expensive to train than shallower neural network architectures. For example, in the experiments the CNN with three convolutional layers takes 685 seconds per epoch[2]. On the other hand, the ResNet with three residual blocks of three convolutional layers each takes 1720 seconds per epoch. In the next subsection, a shallower architecture will be introduced, where the primary aim is to expand the receptive field and gain more access to historic data for the forecasts.

3.7 Residual dilated convolutional neural network

The model introduced in Section 3.5 provides a large receptive field for the forecast of the Linac3 instability. However, a deeper architecture is desirable to compare its performance to a simpler implementation as described in Section 3.5. For this reason, an architecture similar to ResNets, which enables faster and more efficient learning, and dilated convolutions is introduced in this section. More concretely, stacking residual blocks as in Equation (19) is suggested, where the convolutional layers are replaced by dilated convolutions.

The following dilated convolution block is introduced, after allowing dilated convolutions in the block as described in Equation (12)

$$z = K *_d x + b \tag{20a}$$

$$n = \mathrm{BN}(z) \tag{20b}$$

$$h = \mathrm{ReLU}(n) \tag{20c}$$

where $*_d$ is the dilated convolution introduced in Equation (13) with a dilation rate d growing exponentially by 2 and K is the kernel with size 2. Next, three dilated convolutional blocks are connected by a non-linearity and a shortcut connection as in Section 3.6. Hence, a residual block is defined below for three stacked dilated convolution blocks

$$a_1 = \mathrm{dConvBlock}_K(x) \tag{21a}$$

$$a_2 = \mathrm{dConvBlock}_K(a_1) \tag{21b}$$

$$a_3 = \mathrm{dConvBlock}_K(a_2) \tag{21c}$$

$$s = a_3 + x \tag{21d}$$

$$\hat{h} = \mathrm{ReLU}(s) \tag{21e}$$

where each dConvBlock comes from Equation (20), K has size 2 and $s, x, \hat{h}$ have the same functionality as in Equation (19). Last, the dilation rate $d \in [2^0, ..., 2^{L-1}]$ where L denotes the number of dilated layers. For one residual block of three dilated convolution blocks each, the receptive field from Equation (14) is 8.

[2]The experiments have taken place on the Service for Web based ANalysis [44], i.e. SWAN, which is a jyputer-based platform developed at CERN

4 Change point detection methods

Change point detection (CPD) plays an important role in identifying abrupt changes in time series data. CPD has many applications in recognising changes of trend, for example in finance [45]. The objective in this section is to use a CPD method to detect a downward trend in the beam intensity, i.e. BCT25, as quickly and accurately as possible to alert the source specialist. The BCT25 was used in this case to identify changes of trend not only after the source, but also after the Linac as depicted in Figure 5. In general, there are two different types of CPD: online and offline methods. In online methods the entirety of the time series is not known a priori, and hence change points can be detected on streaming data [46]. In offline methods change points can be found only in a full data set [9]. That means that in offline methods, future data points are used to decide whether there is a change point now. In this section, an online and an offline change point detection method, that have been used in this research, will be analysed.

4.1 Direct density-ratio estimation

In this subsection, the notes from Kawahara and Sugiyama will be mostly followed [47].

The direct density-ratio estimation method expands on the idea of identifying change points by calculating the probability density functions that generate the data before and after a target time point. As soon as the two probability density functions are calculated, the log likelihood ratio can be computed for both regions [48]. A threshold μ is applied on the log likelihood ratio, based on which a decision can be made for the existence of a change point.

First, a sample from BCT25 is defined as $\mathbf{x}(t) \in \mathbb{R}$. A subsequence from the BCT25 time series is denoted with $\mathbf{X}(t) \in \mathbb{R}^k$ with length k, where k is a user defined parameter.

$$\mathbf{X}(t) := [\mathbf{x}(t)^\top, \mathbf{x}(t+1)^\top, ..., \mathbf{x}(t+k-1)^\top]^\top$$

The goal is to calculate

$$l(\mathbf{X}) = \ln \frac{p_{\text{after}}(\mathbf{X})}{p_{\text{before}}(\mathbf{X})} \tag{22}$$

with p_{after} and p_{before} denoting the probability density functions before and after a candidate change point in BCT25. Next the exact times where the intervals before and after the change point start are defined as t_{bf} and t_{af}, respectively. The i-th sample before and after a candidate change point in BCT25 are represented with $\mathbf{X}_{bf}(i) = \mathbf{X}(t_{bf} + i - 1)$ and $\mathbf{X}_{af}(i) = \mathbf{X}(t_{af} + i - 1)$, respectively. A sketch is provided in Figure 15 to explain the above notions. For a candidate change point in BCT25 two hypotheses can be made

$$\begin{cases} H_0: & p(\mathbf{X}(i)) = p_{\text{before}}(\mathbf{X}(i)) \text{ for } t_{bf} \leq i < t \\ H_1: & p(\mathbf{X}(i)) = p_{\text{before}}(\mathbf{X}(i)) \text{ for } t_{bf} \leq i < t_{af}, \\ & p(\mathbf{X}(i)) = p_{\text{after}}(\mathbf{X}(i)) \quad \text{for } t_{af} \leq i < t \end{cases} \tag{23}$$

The null hypothesis assumes that no change has occured in the data. On the contrary, the alternative hypothesis assumes that there is a difference in the two probability density functions before and after the change point. Given the two hypotheses it is possible to calculate the likelihood ratio

$$\Lambda = \frac{\prod_{i=1}^{n_{bf}} p_{\text{before}}(\mathbf{X}_{bf}(i)) \prod_{i=1}^{n_{af}} p_{\text{after}}(\mathbf{X}_{af}(i))}{\prod_{i=1}^{n_{bf}} p_{\text{before}}(\mathbf{X}_{bf}(i)) \prod_{i=1}^{n_{af}} p_{\text{before}}(\mathbf{X}_{af}(i))} \tag{24}$$

$$= \frac{\prod_{i=1}^{n_{af}} p_{\text{after}}(\mathbf{X}_{af}(i))}{\prod_{i=1}^{n_{af}} p_{\text{before}}(\mathbf{X}_{af}(i))} \tag{25}$$

where n_{bf} and n_{af} are user defined parameters and provide the number of BCT25 samples that exist in the intervals before and after the candidate change point, respectively. Given Equation (25) the log likelihood ratio L can be derived. With a user defined threshold μ it can be determined whether a change point occured

$$\begin{cases} L \geq \mu, & \text{A change point exists} \\ \text{otherwise}, & \text{No change identified} \end{cases} \tag{26}$$

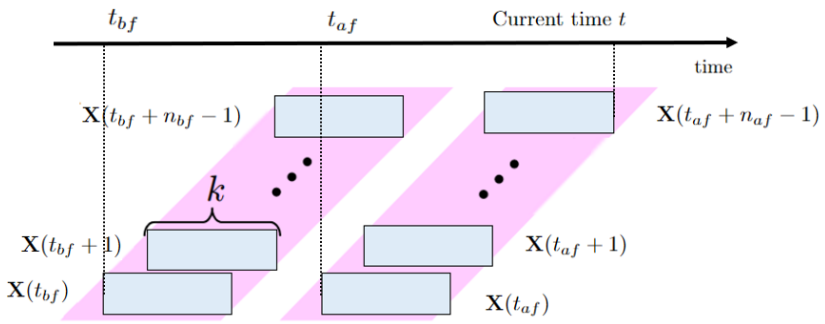


Figure 15: A demonstration of one dimensional time series data, i.e. BCT25, being split into two intervals before and after a candidate change point. Each grey block consists of k time series samples $\mathbf{x}(t)$. Figure motivated from [49].

As stated in [9], estimating the above probability density functions is computationally expensive. Instead, the Kullback-Leibler importance estimation procedure (KLIEP) [50] can be utilised. In fact, literature in [47] indicates that KLIEP performs better than change point detection methods such as *subspace identification* [51] and *one-class support vector machine* [52]. Specifically, KLIEP estimates

$$q(\mathbf{X}) := \frac{p_{\text{after}}(\mathbf{X})}{p_{\text{before}}(\mathbf{X})} \tag{27}$$

with the use of a non-parametric kernel model

$$\hat{q}(\mathbf{X}) = \sum_{i=1}^{n_{af}} \theta_i K_\sigma(\mathbf{X}, \mathbf{X}_{af}(i)) \tag{28}$$

with $\{\theta_i\}_{i=1}^{n_{af}}$ being determined from the BCT25 observations and $K_\sigma(\mathbf{X}, \mathbf{X}_{af}(i))$ being a Gaussian kernel function.

$$K_\sigma(\mathbf{X}, \mathbf{X}_{af}(i)) = \exp\left(-\frac{||\mathbf{X} - \mathbf{X}'||^2}{2\sigma^2}\right) \tag{29}$$

with $\mathbf{X}'$ being the mean and σ denoting the width of the kernel. Specifically, the ratio in Equation (27) can be estimated thanks to the use of the Kullback-Leibler (KL) divergence. Essentially, KLIEP minimises the KL divergence between $p_{\text{after}}(\mathbf{X})$ and $\hat{p}_{\text{after}}(\mathbf{X})$. The latter is equivalent to $\hat{q}(\mathbf{X})p_{\text{before}}(\mathbf{X})$ based on Equation (27). Hence, the KL divergence, which needs to be minimised based on the parameters θ, can be deduced.

$$\text{KL} = \int p_{\text{after}}(\mathbf{X}) \log\left(\frac{p_{\text{after}}(\mathbf{X})}{\hat{q}(\mathbf{X})p_{\text{before}}(\mathbf{X})}\right) \mathrm{d}\mathbf{X} \tag{30}$$

$$= \int p_{\text{after}}(\mathbf{X}) \log\left(\frac{p_{\text{after}}(\mathbf{X})}{p_{\text{before}}(\mathbf{X})}\right) \mathrm{d}\mathbf{X} - \int p_{\text{after}}(\mathbf{X}) \log(\hat{q}(\mathbf{X}))\mathrm{d}\mathbf{X} \tag{31}$$

The first part can be ignored since it has no dependency on the parameter θ. Therefore, the following optimisation problem is formulated

$$\begin{aligned} \underset{\{\theta_j\}_{j=1}^{n_{af}}}{\text{maximize}} \quad & \sum_{i=1}^{n_{af}} \log\left(\sum_{j=1}^{n_{af}} \theta_j K_\sigma(\mathbf{X}_{af}(i),\ \mathbf{X}_{af}(j))\right) \\ \text{subject to} \quad & \frac{1}{n_{bf}} \sum_{i=1}^{n_{bf}} \sum_{j=1}^{n_{af}} \theta_j K_\sigma(\mathbf{X}_{bf}(i), \mathbf{X}_{af}(j)) = 1, \\ & \theta_1, ..., \theta_{n_{af}} \geq 0 \end{aligned} \tag{32}$$

The optimisation problem in Equation (32) is convex and therefore a global maximum $\hat{\theta}$ is achieved [49]. The density-ratio can now be estimated

$$\hat{q}(\mathbf{X}) = \sum_{i=1}^{n_{af}} \hat{\theta}_i K_\sigma(\mathbf{X}, \mathbf{X}_{af}(i)) \tag{33}$$

and hence estimate the log likelihood ratio

$$\hat{L} = \frac{1}{n_{af}} \sum_{i=1}^{n_{af}} \log \hat{q}(\mathbf{X}_{af}(i)) \tag{34}$$

where with the use of threshold μ, as mentioned in Equation (26), a decision whether a change point exists can be made. An example is provided in Figure 16.

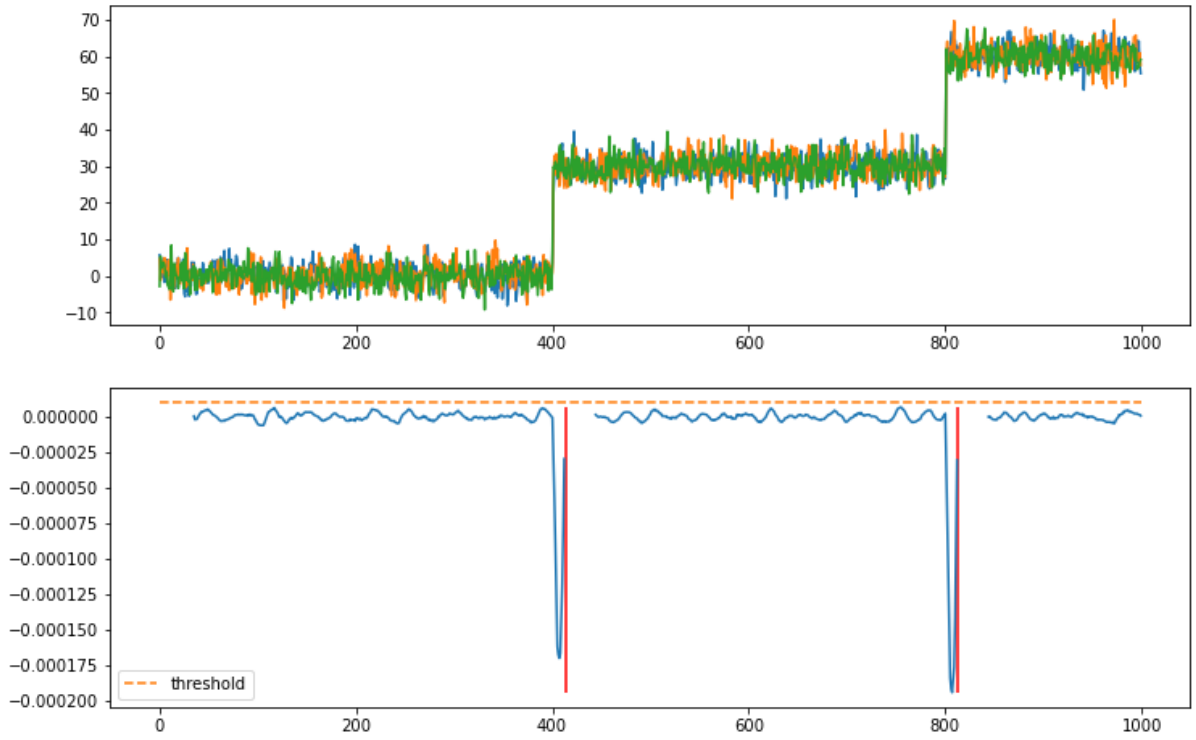


Figure 16: Example of change point detection on simulated time series (top) using the KLIEP package [53]. (bottom) Likelihood ratios are plotted with threshold (orange) being set to $\mu = 0.00001$ and change points denoted with a red vertical line.

It is observed that KLIEP requires many user defined parameters, i.e. k, n_{af}, n_{bf}, μ, σ, which are difficult to tune simultaneously given the running time of the change point detection algorithm on Linac3 data. This motivates the current research to apply another change point detection algorithm which is more computationally efficient and less cumbersome for parameter tuning.

4.2 Bayesian online change point detection

In this subsection, the notes from Adams and MacKay [54] and Küchler and Mihailescu [55] will be followed.

In Bayesian statistics it is assumed that a random vector $\mathbf{x} = (x_1, ..., x_t)$, following a probability distribution $\mathcal{D}$, depends on an unknown parameter θ which is regarded a random variable. Due to this dependency, $P(\theta|\mathbf{x})$ gives the conditional density of θ given $\mathbf{x}$. With the use of the Bayes theorem it follows

$$P(\theta|\mathbf{x}) = \frac{P(\mathbf{x}|\theta)P(\theta)}{P(\mathbf{x})} \propto P(\mathbf{x}|\theta)P(\theta) \tag{35}$$

Since θ is also perceived as a random variable, $P(\theta)$ is the probability distribution from which θ is produced. It is also known as the prior distribution. What is more, $P(\mathbf{x}|\theta)$ is known as the likelihood, which refers to the probability of observing the data $\mathbf{x}$ based on the assumption for parameter θ. Last, $P(\theta|\mathbf{x})$ is known as the posterior distribution, which indicates the knowledge regarding parameter θ given newly observed data $\mathbf{x}$. These notions will be used to explain the algorithm analysed below.

The Bayesian online change point detection (BOCD) method proposed in [54] was used to identify abrupt changes in the BCT25 measurements. It is based on the idea of the run length r_t, meaning how long it was since a change point occured. For example, in case where $r_t = 5$ it suggests that a change point took place at time $t - 5$. What is more, the run length for time step t, i.e. r_t, can be written in the following way

$$r_t = \begin{cases} 0 & \text{A change point exists at time } t \\ r_{t-1} + 1 & \text{No change point exists at time } t \end{cases}$$

The goal of BOCD is to estimate the run length distribution of the beam intensity given all the previous data points, denoted by the posterior distribution $P(r_t|x_{1:t})$, where $x_{1:t} = x_1, ..., x_t$. In Figure 17 an example of a 1-dimensional time series data is given with its corresponding run length.

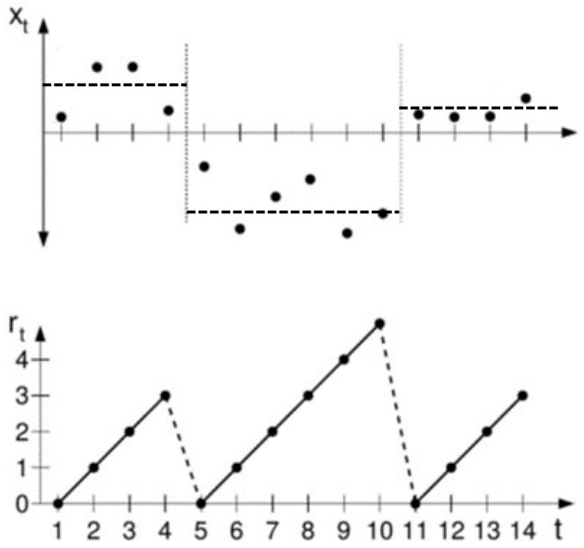


Figure 17: Figure motivated from [54]. (top) It can be spotted that the time series are split into 3 segments due to the change points occuring at times $t = 5$ and $t = 11$ respectively. (bottom) Plot of run length r_t over time. The run length is 0 in the case of a change point.

The posterior distribution can also be written as

$$P(r_t|x_{1:t}) = \frac{P(r_t, x_{1:t})}{P(x_{1:t})} \tag{36}$$

the numerator in Equation (36) can be written in the following way from [56]

$$P(r_t, x_{1:t}) = \sum_{r_{t-1}} P(r_t|r_{t-1}) P(x_t|r_{t-1}, x_t^{(l)}) P(r_{t-1}, x_{1:t-1}) \tag{37}$$

where $r_{t-1} = l$ and $x_t^{(l)} = (x_{t-l}, ..., x_{t-1})$. As soon as $P(r_{t-1}, x_{1:t-1})$ is determined, it can be forward message-passed to work out $P(r_t, x_{1:t})$. The first part in Equation (37) is the change point prior and gives insight to the model about the change points observed in the data. The assumption that is made for the change point prior is the following

$$P(r_t|r_{t-1}) = \begin{cases} H(r_{t-1} + 1) & \text{if } r_t = 0 \\ 1 - H(r_{t-1} + 1) & \text{if } r_t = r_{t-1} + 1 \\ 0 & \text{else} \end{cases} \tag{38}$$

where H is the Hazard function providing the likelihood of an sudden change occurring for a run length r_t. In the case of Linac3, the change points are assumed to be geometrically distributed. What is more, the expected distance between two abrupt changes is set to $\mu = 6$ hours. The value of μ was determined after implementing this change point detection method multiple times on BCT25 data. Given the above assumptions, the Hazard function is $H(y) = \frac{1}{\mu}$.

The second part of Equation (37) corresponds to the likelihood that BCT25 at time t is x_t, whilst at r_{t-1} the BCT25 was $x_t^{(l)}$. Thus, only the last l data points are taken into consideration. In the context of Linac3, the source specialists assume that BCT25 is following a normal distribution. Therefore, it follows that $x_t|r_{t-1}, x_t^{(l)} \sim N(\mu, \sigma^2)$ with μ, σ^2 being parameters to be determined. However, since there is variability about the estimation of μ, σ^2, the following weighted average distribution is applied

$$P(x_t|r_{t-1}, x_t^{(l)}) = \int_\theta P(x_t|\theta)P(\theta|r_{t-1}, x_t^{(l)})\mathrm{d}\theta \tag{39}$$

$$= \int_{\mu,\sigma^2} P(x_t|\mu, \sigma^2)P(\mu, \sigma^2|r_{t-1}, x_t^{(l)})\mathrm{d}(\mu, \sigma^2) \tag{40}$$

The first part explains how likely the given observation is and the second part is the prior indicating how well the parameter describes the previous observations. For the computation of Equation (40) the conjugate prior is utilised [57]. Here, the first part is normally distributed which is in agreement with the assumption about BCT25 data. From the BCT25 data, the first hour of observations can be taken to calculate the sample mean and variance, i.e. $(\hat{\mu}, \hat{\sigma}^2)$. The conjugate prior from [58], is proven to be the Normal Inverse Gamma Distribution and therefore

$$\sigma^2|\alpha, \beta \sim \text{InverseGamma}(\alpha, \beta) \tag{41}$$

$$\mu|\hat{\sigma}^2, \hat{\mu}, \lambda \sim \text{Normal}\left(\hat{\mu}, \frac{\hat{\sigma}^2}{\lambda}\right) \tag{42}$$

where λ is the number of observations in 1 hour of data to estimate the mean $\hat{\mu}$, α is the number of observations in 30 minutes of data to estimate the variance $\beta = \alpha\hat{\sigma}^2$. Also from [58] it follows

$$x_t|r_{t-1}, x_t^{(l)} \sim t_{2\alpha}\left(\hat{\mu}, \frac{\hat{\beta}(\lambda+1)}{\alpha\lambda}\right) \tag{43}$$

where t is a t-distribution and 2α is a parameter which denotes the degrees of freedom. The parameters $\mu, \alpha, \lambda, \beta$ get updated for every BCT25 datapoint

$$\lambda' \longleftarrow \lambda + 1 \tag{44}$$

$$\alpha' \longleftarrow \alpha + \frac{1}{2} \tag{45}$$

$$\mu' \longleftarrow \frac{\lambda\mu + x}{\lambda + 1} \tag{46}$$

$$\beta' \longleftarrow \beta + \frac{\lambda}{\lambda + 1}\frac{(x-\mu)^2}{2} \tag{47}$$

From prior research done on LINAC3 [55], a change point can be found using the Maximum a posteriori estimate (MAP) [59, 60]. MAP can be seen as a way to segment the BCT25 data in the most optimal way by finding the most representative change point to indicate a beam decay. First, the possible change points are examined and subsequently the ones with the highest probability based on the current BCT25 measurements are chosen. An example of using BOCD on time series data is provided in Figure 18.

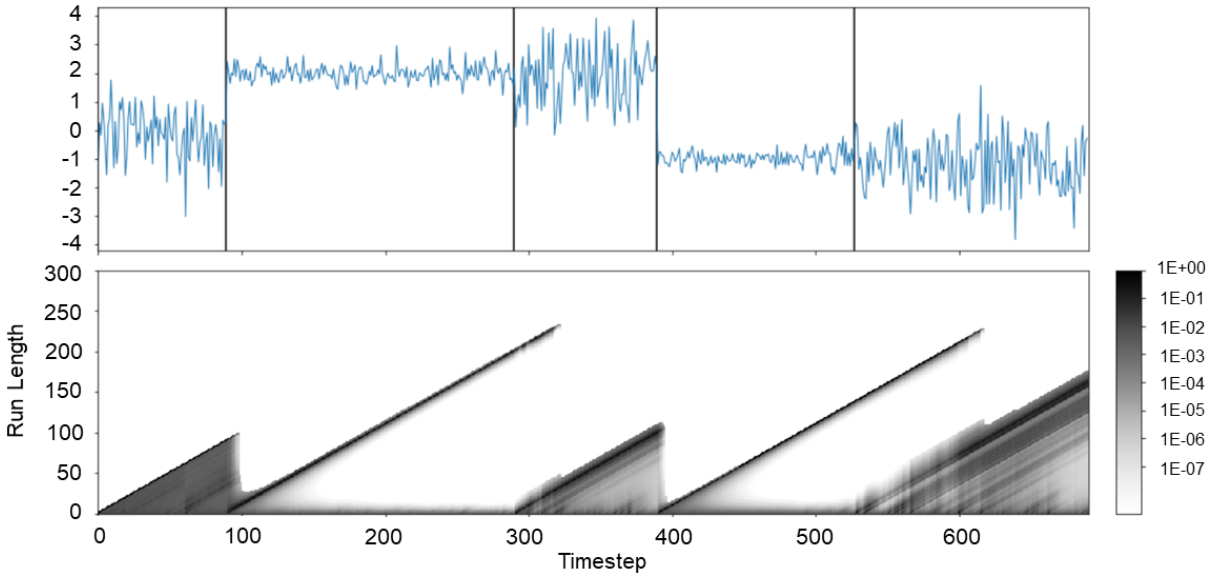


Figure 18: (top) Example of change point detection on simulated time series using BOCD. Change points are depicted with a black vertical line. (bottom) Run length probabilities are plotted. The plot is darker when the probability increases.

To improve the computation time of the algorithm, any probability for the run length distribution which is less than 10^{-4} is set to zero [54]. In comparison to KLIEP, BOCD computes change points faster. Most importantly the only parameters which need to be set beforehand are λ and α, and hence, BOCD is easier to implement.

5 Results

In this section, the results from both forecasting and change point detection methods will be discussed. First, the measures with which the performance of the forecasting models was evaluated, will be explored in Section 5.1. Next, the architectures of the neural networks and the kernels used for SVMs that were used in the experiments will be described in more detail in Section 5.2. The results from the models will be outlined and discussed in Section 5.2.1. Last, the outcome from the implementation of the change point detection methods will be shown in Section 5.3.

5.1 Model performance measures

In classification problems, a way to describe and visualise the predictive ability of a classification model is to plot the confusion matrix [61]. The confusion matrix used in this research that summarises the results of the forecasting is demonstrated in Figure 19.

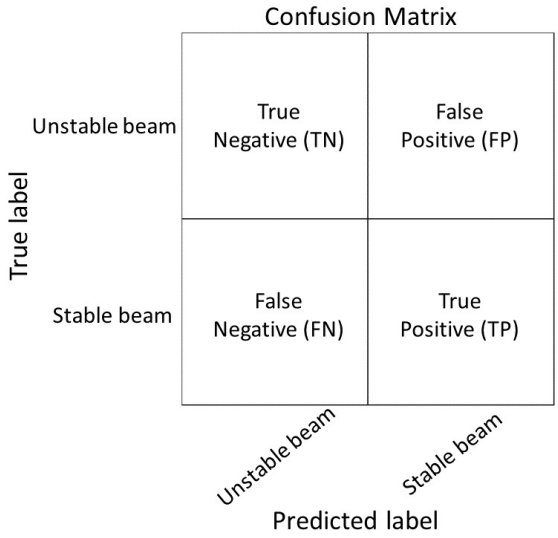


Figure 19: An example of the confusion matrices that will be used in this analysis. The correctly predicted instabilities and stabilities are the true negatives and true positives respectivelly. A wrongly predicted instability is a false negative whist a wrongly predicted stability is a false positive.

This matrix gives an understanding of how well each forecasting method performs by providing the number of correctly predicted stabilities/instabilities, i.e. the number of true positives and true negatives respectively, and mislabellings, i.e. the number of false positives and false negatives. Additionally, it gives insight in the case where one label is predicted more accurately from the other. This is typically possible in the case of imbalanced data, where a classifier is more biased to predict the majority class. In this research, the train dataset for the forecasting models, starting on the 1st until the 25th of August 2021, has 83% of its observations being labelled as stable. Therefore, the labels in Linac3 data are imbalanced.

Moreover, the accuracy metric, from Equation (48a), is used to measure the performance of a classifier. That is because Tensorflow/Keras [62, 63], which was utilised for the forecasting with the neural networks, uses this metric during the training and validation phase. However, in imbalanced data the accuracy metric can be misleading. For example, in the case where a neural network classifies every observation as stable in the training data, it would yield 83% accuracy although it failed completely forecasting any instability. For this reason, the True positive rate (TPR) and False positive rate (FPR) metrics are also introduced in Equation (48b) and Equation (48c) respectively

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{48a}$$

$$\text{True Positive Rate} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{48b}$$

$$\text{False Positive Rate} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{48c}$$

Receiver Operating Characteristic (ROC) curve

Another method used to visualise the performance of the forecasting models is the Receiver Operating Characteristic (ROC) curve [64]. The ROC curve allows more than one metric, i.e. TPR and FPR, to assess the overall ability of the classifier. This is depicted in Figure 20.

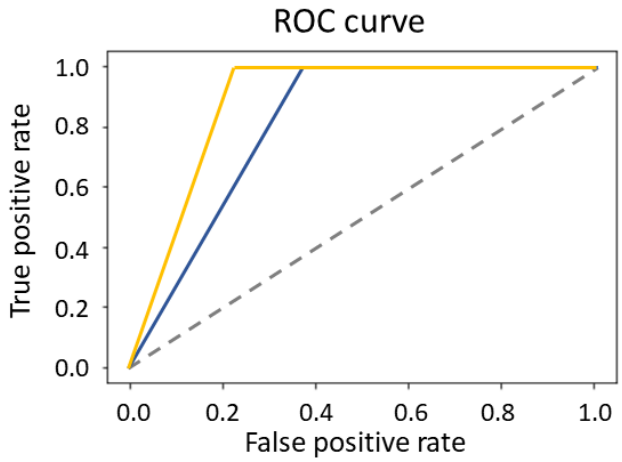


Figure 20: An example of the ROC curve comparing the predictive ability of two classification models. Here, the yellow model has provided better predictions. The dotted line denotes a random classifier.

A perfect classifier is expected to have a curve passing through the points $(0, 0), (0, 1), (1, 1)$. Specifically, when TPR $= 1$ and FPR $= 0$, the classifier has neither false positives nor false negatives. Moreover, for probabilistic classifiers, such as the neural networks introduced in Section 3, the ROC curve calculates the pair (FPR, TPR) for a wide range of probability thresholds arbitrarily set from `scikit-learn` [65]. Given a probability threshold, e.g. 0.5, the classifier decides to forecast a beam stablility or instability based on historic observations of the features provided in Table 2. Based on the ROC curve, all forecasting models will be compared. Last, the Area Under the Curve (AUC) quantifies the overall ability of the model by calculating the area under the ROC curve. Hence, a perfect classifier would have AUC $= 1$.

5.2 Forecasting experiments

The models in Section 3 were trained with the data of the first 3 weeks of August 2021 and validated using the last week of August to provide predictions with a 15 and 60 minute forecast horizon respectively. During the validation phase, the hyperparameters of each model were determined based on its performance. To avoid overfitting on the neural networks, early stopping was applied in Keras. This allows the neural networks to stop training when the validation accuracy stops increasing. Moreover, the Adaptive moment estimation (Adam) optimisation algorithm was used to optimise the cost function of the neural networks [66]. Finally, the tuned models with the best performance in the validation set were subsequently tested on unseen data, i.e. October 2021.

Bayesian optimisation was implemented for the choice of hyperparameters in each neural network, such as the number of nodes in a dense layer, kernel size, the number of filters in each convolution, dropout rate and learning rate [67, 68, 69]. The Bayesian hyperparameter optimisation allows searching in a range of values for each parameter, however, in contrast to other methods such as random search or grid search, each iteration takes into consideration the results from the previous iterations. In this way, the next choice of hyperparameters is based on informed decisions and it can find a good range, where the model achieves higher accuracy on unseen data. The final tuning of each neural network model was determined after 30 different initialisations by choosing the hyperparameters that yield the highest accuracy in the validation set. The search ranges are provided in the Appendix A.

MLP hyperparameters

For the 15 minute forecast, the MLP model after the hyperparameter tuning is consisted of 3 dense layers of 250 ReLU nodes each followed by a dropout layer with rate 0.1, 0.1, 0.15 respectively. For the 60 minute forecast, the MLP model has 3 dense layers of 400 ReLU nodes, each followed by a dropout

layer with rate 0.2. By increasing the number of layers, none of the two models increased the accuracy in the validation set.

CNN hyperparameters

For the CNN models, both consist of 3 convolutional blocks as described in Equation (12). For the 15 minute forecast, each convolutional block has 120, 240, 120 filters with kernel size 10, 5, 2 respectively. For the 60 minute forecast, the convolutional blocks have 120, 240, 100 filters each and kernel size 8, 4, 2 respectively.

ResNet hyperparameters

For the 15 minute forecast, the ResNet is comprised of 3 residual blocks as indicated in Equation (19), where the first block has 100 filters and the other two have 200 filters with kernels of size 15, 9, 2 each. In the 60 minute forecast, the ResNet has 80 filters for the first block where the other two have 160 filters and kernels of size 40, 20, 10.

Dilated CNN hyperparameters

This model, for the 15 minute forecast, consists of 9 dilated convolutional layers with 100 filters and kernel size 2 each followed by a dense layer of 200 ReLU nodes and a dropout layer with rate 0.1. The 60 minute forecast model has 9 dilated convolutional layers with 85 filters and kernel size 2 with dilation rate $d \in [2^0, ..., 2^8]$. It is the followed by a dense layer of 280 ReLU nodes and dropout rate 0.3.

Residual dilated CNN hyperparameters

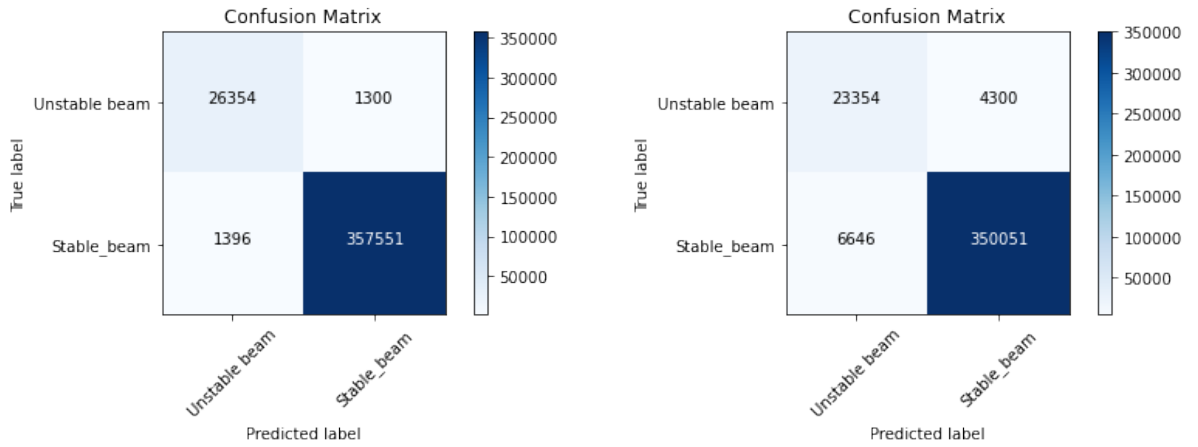
As discussed in Section 3.7, the 15 minute forecast model consists of 3 residual blocks of 3 dilated convolutional layers, each with 100 filters on the first block, 200 filters on the other two and kernel size 2 with dilation rate $d \in [2^0, ..., 2^8]$. The 60 minute forecast model has 67 filters on the first residual block and 134 on the other two.

SVM parameter tuning

Grid search [70] was implemented for the choice of the right kernel for the SVM model. Specifically, the SVM models were evaluated on the validation set for four possible kernels: linear, polynomial, rbf and sigmoid. Moreover, the models were evaluated for different values of ℓ_2-regularisation penalty C, which is available from `scikit-learn`. Based on these tests, the sigmoid kernel with regularisation penalty $C = 0.1$ provided the highest validation accuracy. Last, different class weights were assigned on each label, given that Linac3 data are imbalanced. In this way, the algorithm penalises more misclassifications regarding an instability, i.e. label 0. For the 15 minute forecast, the model with the highest validation accuracy assigned 10 times more weight to label 0 than label 1. For the 60 minute forecast, the model with the highest validation accuracy allocated the same weight for both labels.

5.2.1 Forecasting Results

The persistence model as described in Section 3.1 is the baseline model, which the other models need to outperform in order to conclude that a beam instability is possible to forecast. The performance of the persistence model for both forecast horizons is presented in Figure 21.



(a) 15 minute forecast (99.3% accuracy)

(b) 60 minute forecast (97.2% accuracy)

Figure 21: Persistence model in the test dataset.

Since the output layer of all neural networks is a softmax layer, a probability threshold of 0.5 was set to determine a stability or instability with labels 1 and 0 respectively. In the conducted experiments for the 15 and 60 minute forecast, the MLP model was first implemented. For the 15 minute forecast, the confusion matrices of two models are illustrated in Figure 22a and Figure 22b. Likewise, for the 60 minute forecast the confusion matrices are provided in Figure 23a and Figure 23b. The models are more inclined to make mistakes when forecasting an instability, i.e. false negatives. This was also observed in the validation set. To remedy this, different class weights were assigned manually for each label in Keras. However, this action deteriorated the overall validation accuracy of the model and subsequently this idea was never implemented on the test dataset. In addition, the ROC curve, provided in Figure 22c and Figure 23c, compares two models yielded from the Bayesian hyperparameter tuning with the persistence model for the 15 minute and 60 minute forecast respectively. From these results, it can be concluded that the MLP model is not able to exceed the performance of the persistence model. This lack of performance can be attributed to the fact that the MLP model assigns a different weight to every observation. As a result, the MLP considers every data point as independent from the previous one.

The next model that was tested on unseen data was the CNN model. Unlike the MLP model, the CNN model is able to take into account the temporal information provided in the data by sharing the weights in every kernel. As a result, the model performs better than the MLP on unseen data for both forecast horizons. In particular, the confusion matrices of two CNN models for the 15 minute forecast are provided in Figure 24a and Figure 24b for a probability threshold 0.5. Subsequently, the models are compared with the persistence model on the ROC curve in Figure 24c for a wider range of thresholds. Similarly, the 60 minute binary forecasts for two different sets of hyperparameters are represented in Figure 25a and Figure 25b. Moreover, the ROC curve for the 60 minute forecast is provided in Figure 25c. From both ROC curves, it can be concluded that the CNN model is unable to capture the necessary patterns to effectively forecast a decay compared to persistence.

The ResNet model was applied to introduce a deeper neural network architecture that enhances the training process through the residual blocks. During the validation phase, the ResNet was able to always outperform the MLP and CNN models for both forecast horizons. It is therefore surprising that the ResNet only matches the performance of the CNN in the 15 minute forecast on the test dataset, although the same number of validation trials was used during the hyperparameter tuning of both models. This could be attributed to the fact that the wider receptive field of the ResNet was not needed for a short-term forecast in the test dataset. A comparison of two ResNet models with persistence is provided in Figure 26c. The confusion matrices of these two models are additionally illustrated in Figure 26a and Figure 26b. It is highly interesting to see two different models achieving similar overall performance in Figure 26c whilst being biased towards forecasting stability and instability respectively. The same phenomenon appears for the 60 minute forecast in Figure 27a and Figure 27b. However, the ResNet with the highest forecasting ability in the 60 minute forecast is the model that has more bias towards stability (Figure 27c). For both forecasts, the ResNet was not able to surpass persistence.

The Dilated CNN presented the idea of increasing the range of historical data that influence the decision on each forecast. The number of dilated convolutional layers was decided based on the accuracy of

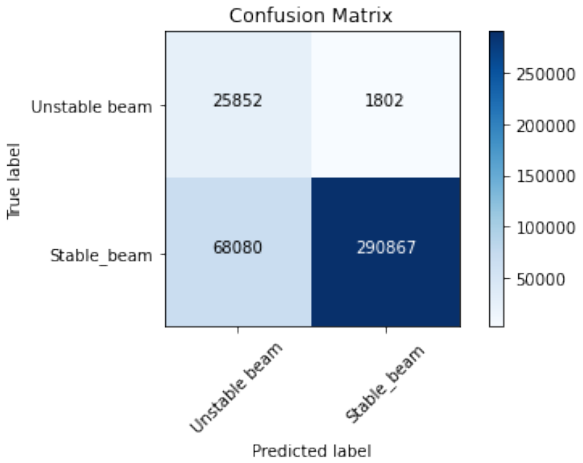
the forecast in the validation set. In Figure 28a and Figure 29a, the results of a model before further hyperparameter tuning has taken place are shown. Despite the fact that there are no signs of overfitting in the validation set, the gap between the accuracy in the validation and test set is significant. After tuning the model for the 15 and 60 minute forecast in Figure 28b and Figure 29b, the model's performance on the test set increases notably. The ROC curve for both forecasts can be seen in Figure 28c and Figure 29c. The results compared to models such as MLPs indicate that taking into account a wider history of data with the given neural network architecture is worse than taking every observation independently to make a forecast. This assumption will be investigated with the next Residual dilated neural network. Another possible extension of the dilated CNN model would be to also implement a batch normalisation layer. This could influence the ability of the model to generalise better on the test set. However, in no experiment the dilated CNN has shown better forecasting performance than persistence.

Throughout the experiments in the validation set, the Residual dilated neural network has been the model with the highest accuracy for the 60 minute horizon forecast. The results of the model are shown in Figure 31a and Figure 31b. Nevertheless, it yields exactly the same forecasting performance with the persistence model in Figure 31c on test data. In this way, it can be concluded that a wide range of historical data is preferable than taking each observation individually, but is more effective for longer forecast horizons, i.e. 60 or 100 minute forecast. On the other hand, this can be confusing for a shorter term forecast such as 15 minutes. The performance of two models for the 15 minute forecast are shown in Figure 30a, Figure 30b and compared to persistence in Figure 30c. By increasing the number of residual blocks, persistence could be reached but not surpassed.

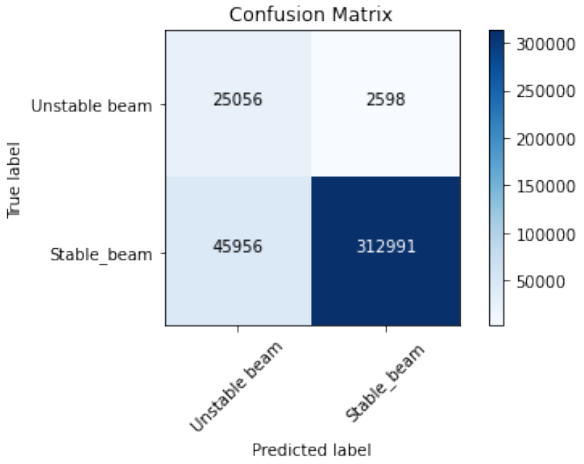
For completeness, a classical ML model (SVM) was tested to forecast a beam decay. During the validation phase in the 15 minute forecast, the model with the sigmoid kernel provided validation accuracy comparable to the MLP models. However, when the same models were applied on the test dataset, they failed to generalise. This is depicted in Figure 32a and Figure 32b. A comparison with the persistence model is available in Figure 32c. Similarly, the SVM showed poor performance on the test data for the 60 minute forecast in Figure 33a and Figure 33b. A summary of these results is also seen in Figure 33c.

Overall, for the 15 minute forecast, the neural networks introducing convolutions with a small receptive field, i.e. ResNet and CNN, have shown the best performance on unseen data. This was expected since the forecast, for such short horizon, does not depend on a wide range of historical data as also seen in previous work done by the Beams and Industrial Control Systems group at CERN. This is also confirmed from observing that the Residual dilated CNN, which is built to look deep into the past, and the MLP, which disregards any temporal information, have similar performance. For this reason, the dilated CNN also underperformed in the test data. The performance of the SVMs until the validation set indicated them as a strong forecasting model compared to other models, such as the dilated CNN and MLP. However, they have shown the worst performance due to their lack of generalisability. A synopsis of all models for the 15 minute forecasts is shown in Figure 34a.

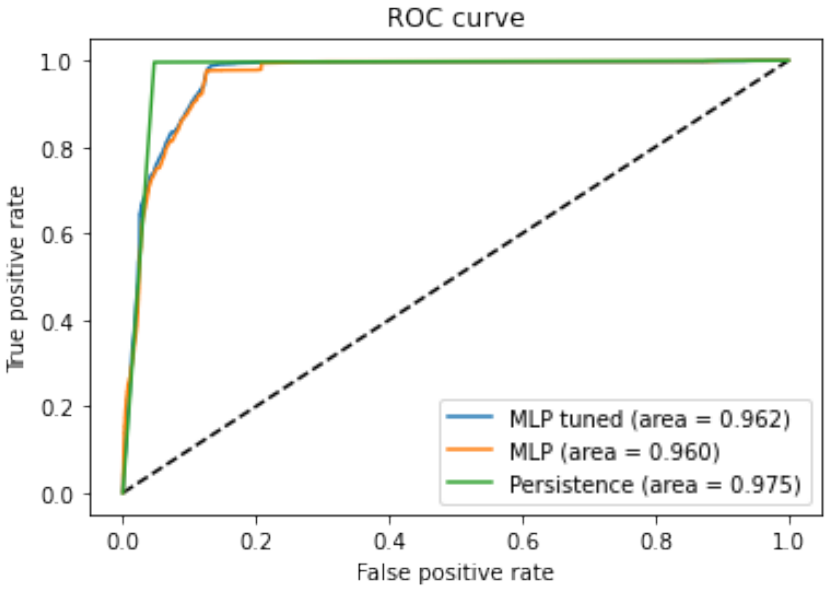
Finally, in the 60 minute forecast, the residual dilated CNN was the only model able to reach exactly persistence in the test data. The convolutional neural network architectures provided better forecasts than MLPs and SVMs, indicating that temporal dependencies might exist on Linac3 data that could be explored to provide a better long term forecast. Nevertheless, the convolutional neural network architectures are not enough to exceed the baseline model. An outline of the 60 minute forecast is given in Figure 34b.



(a) Test performance of the final MLP architecture.

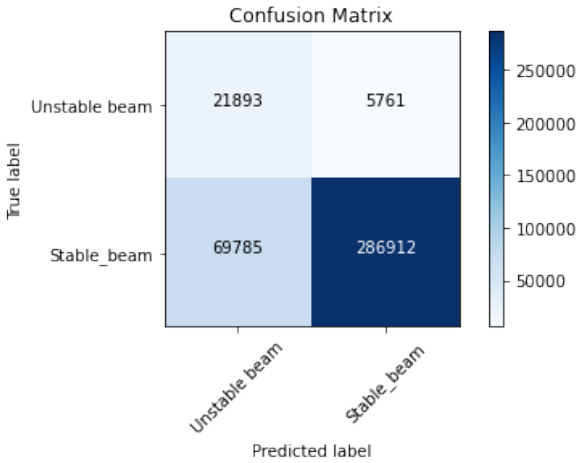


(b) Best MLP model from Bayesian hyperparameter tuning.

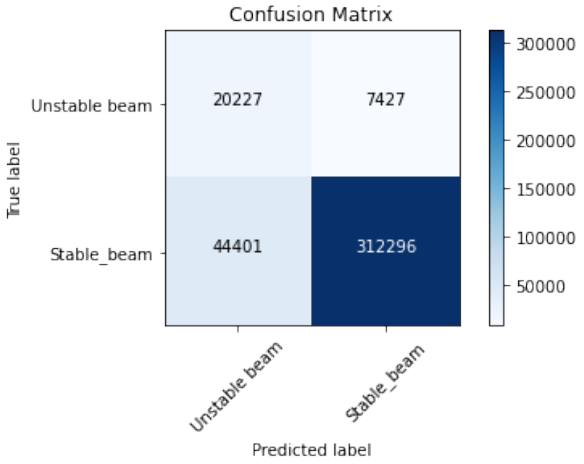


(c) Comparison of the two aforementioned MLP models with the persistence model.

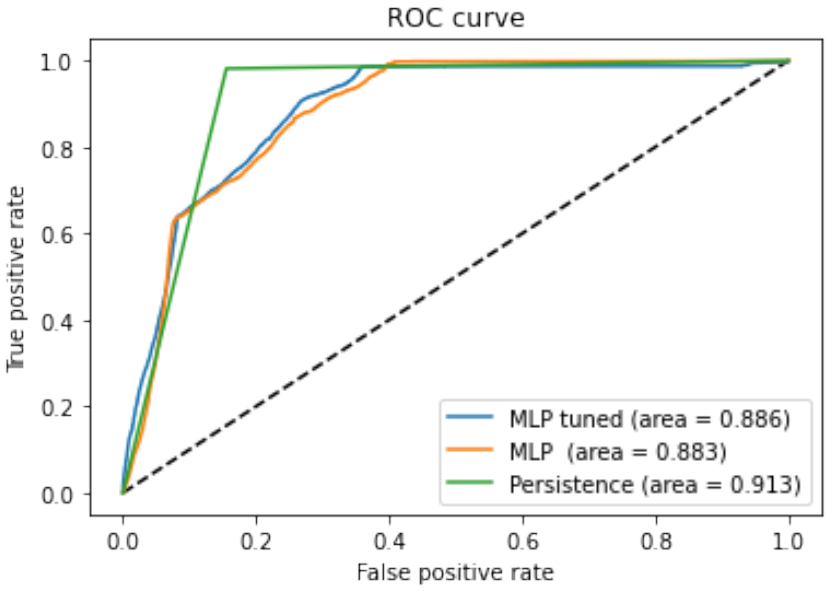
Figure 22: MLP model results for the 15 minute forecast on the test dataset.



(a) Test performance of the final MLP architecture.

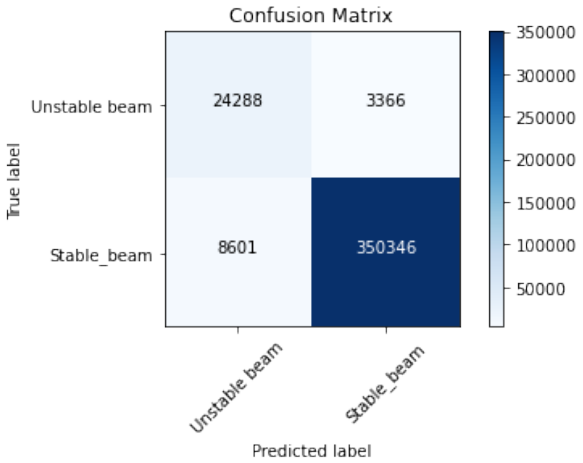


(b) Best MLP model from Bayesian hyperparameter tuning.

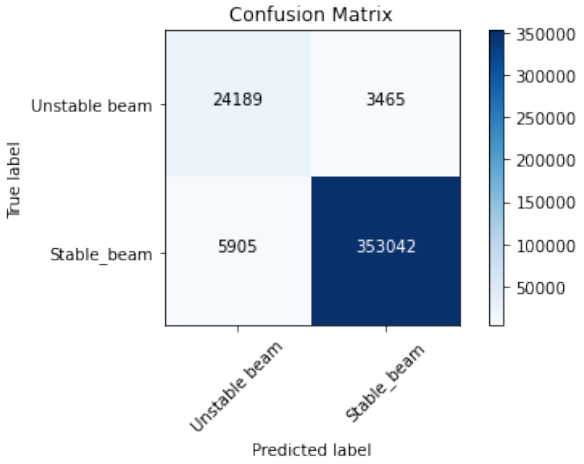


(c) Comparison of the two aforementioned MLP models with the persistence model.

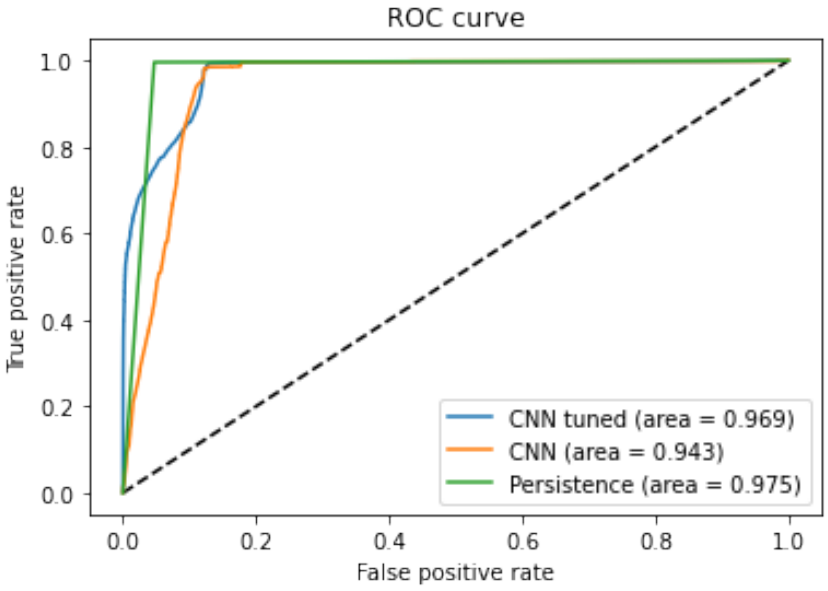
Figure 23: MLP model results for the 60 minute forecast on the test dataset.



(a) Test performance of the final CNN architecture.

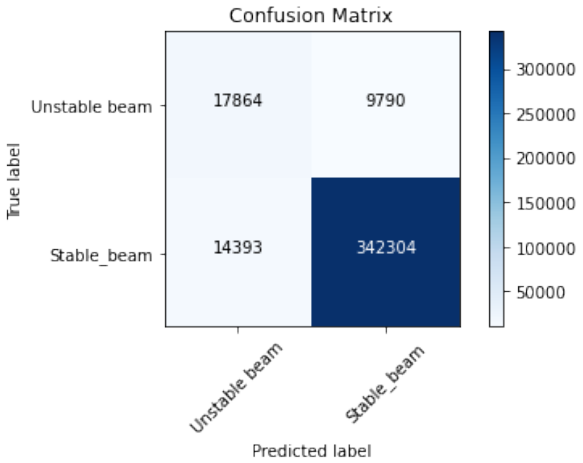


(b) Best CNN model from Bayesian hyperparameter tuning.

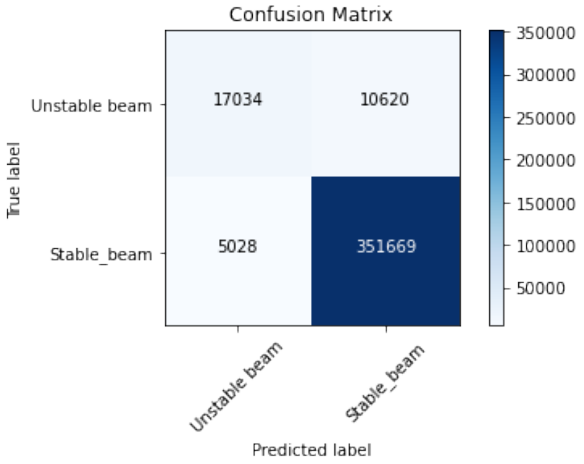


(c) Comparison of the two aforementioned CNN models with the persistence model.

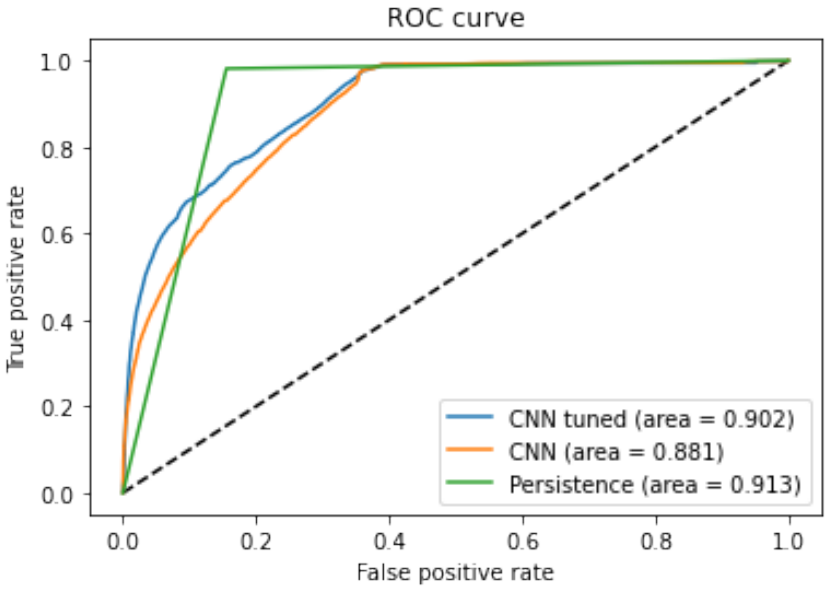
Figure 24: CNN model results for the 15 minute forecast on the test dataset.



(a) Test performance of the final CNN architecture.

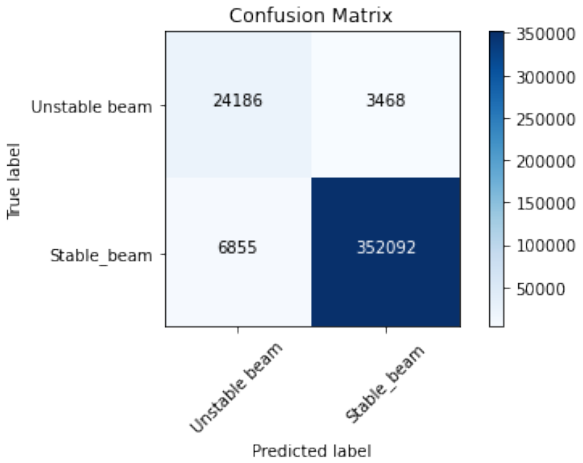


(b) Best CNN model from Bayesian hyperparameter tuning.

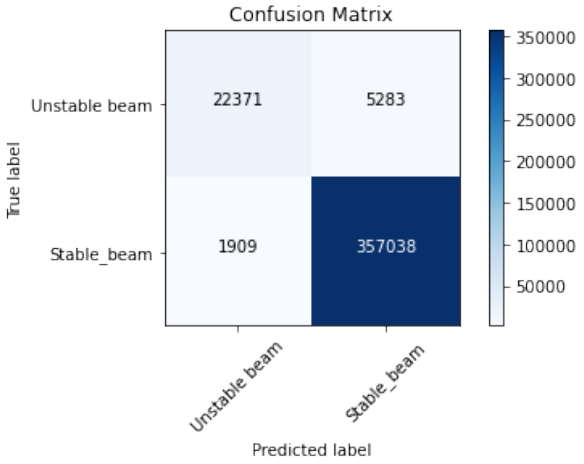


(c) Comparison of the two aforementioned CNN models with the persistence model.

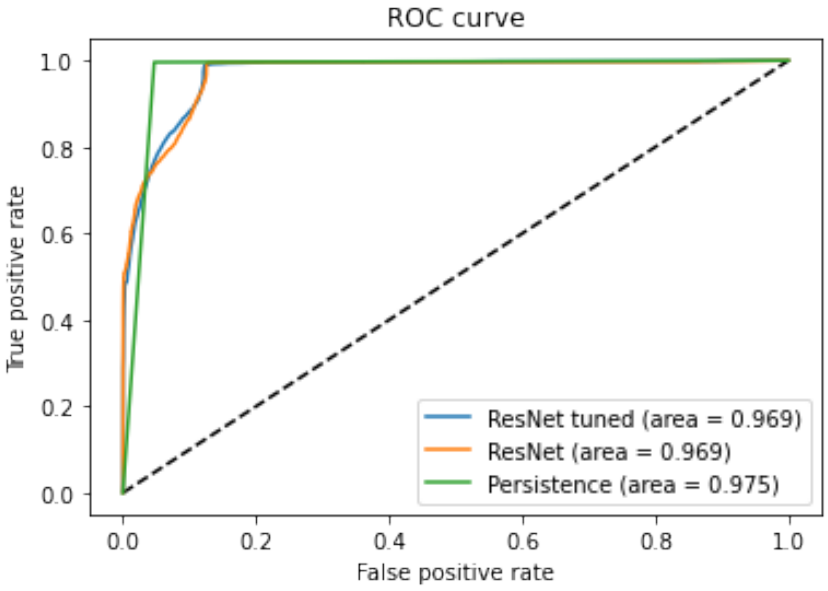
Figure 25: CNN model results for the 60 minute forecast on the test dataset.



(a) Test performance of the final ResNet architecture.

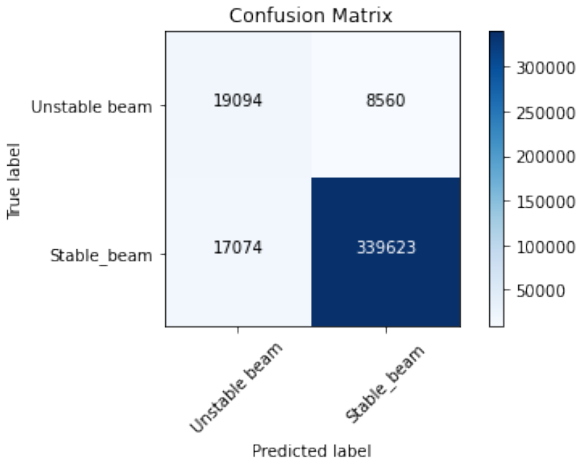


(b) Best ResNet model from Bayesian hyperparameter tuning.

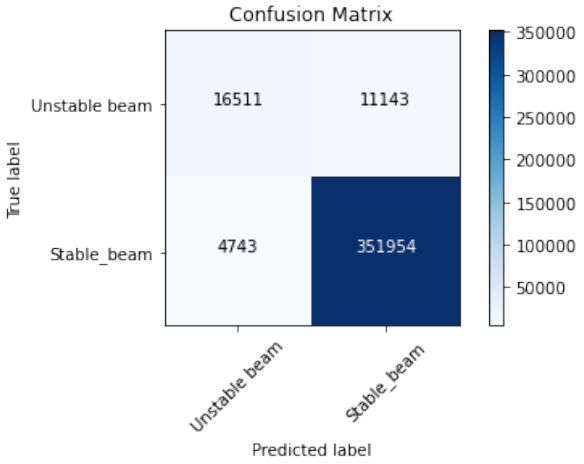


(c) Comparison of the two aforementioned ResNet models with the persistence model.

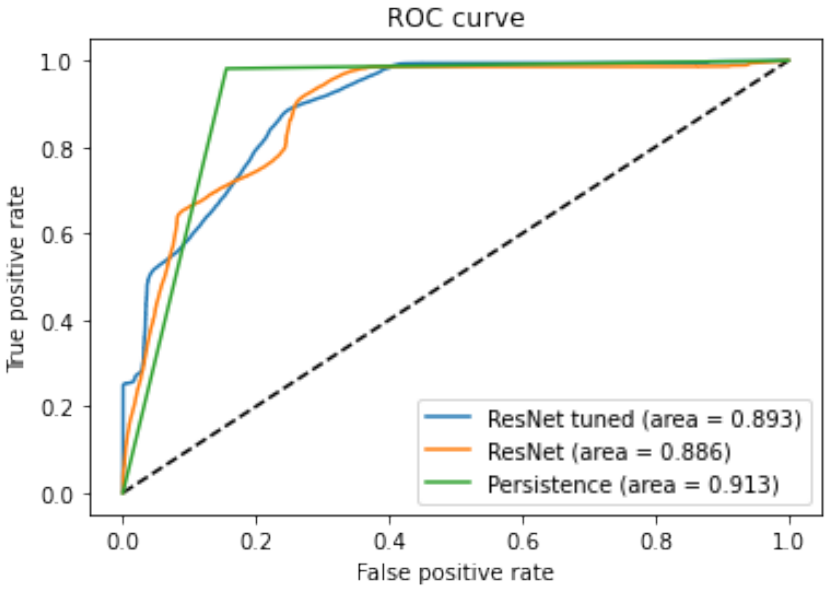
Figure 26: ResNet model results for the 15 minute forecast on the test dataset.



(a) Test performance of the final ResNet architecture.

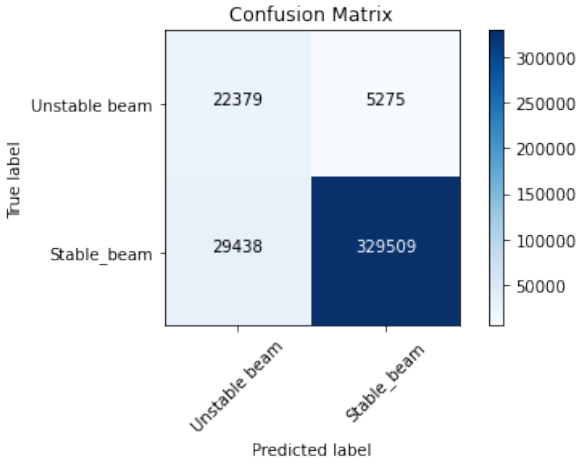


(b) Best ResNet model from Bayesian hyperparameter tuning.

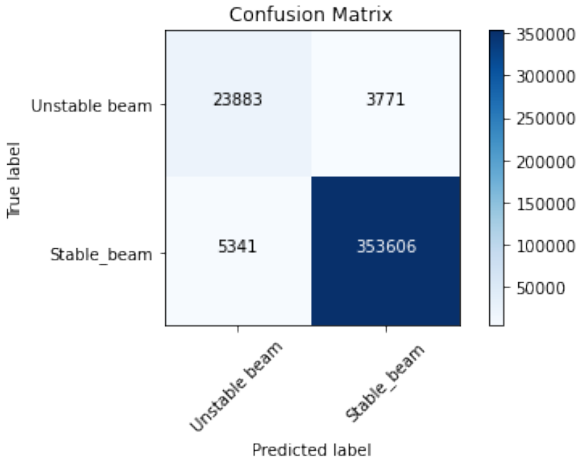


(c) Comparison of the two aforementioned ResNet models with the persistence model.

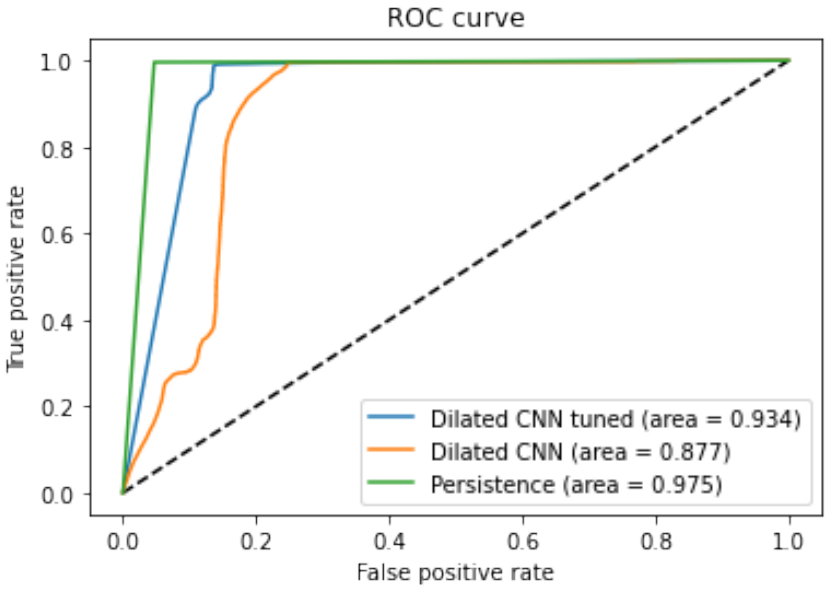
Figure 27: ResNet model results for the 60 minute forecast on the test dataset.



(a) Test performance of the final Dilated CNN architecture.

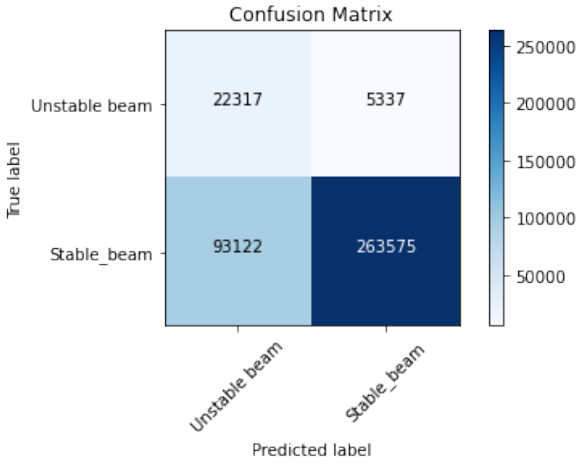


(b) Best dilated CNN model from Bayesian hyperparameter tuning.

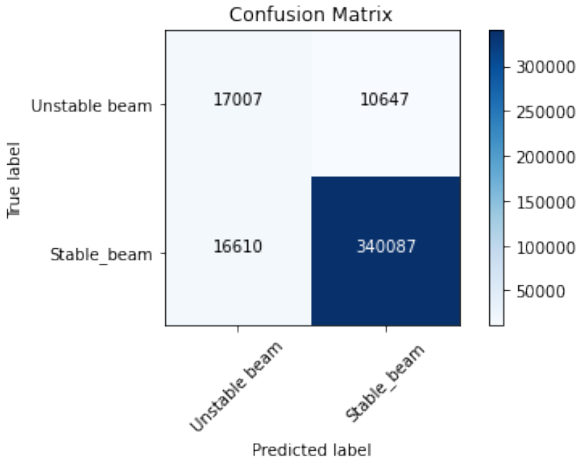


(c) Comparison of the two aforementioned Dilated CNN models with the persistence model.

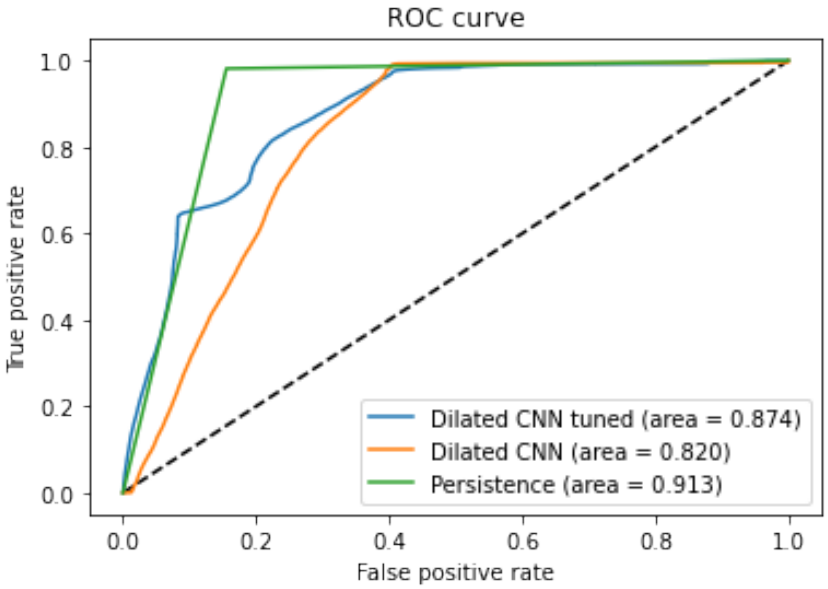
Figure 28: Dilated CNN model results for the 15 minute forecast on the test dataset.



(a) Test performance of the final Dilated CNN architecture.

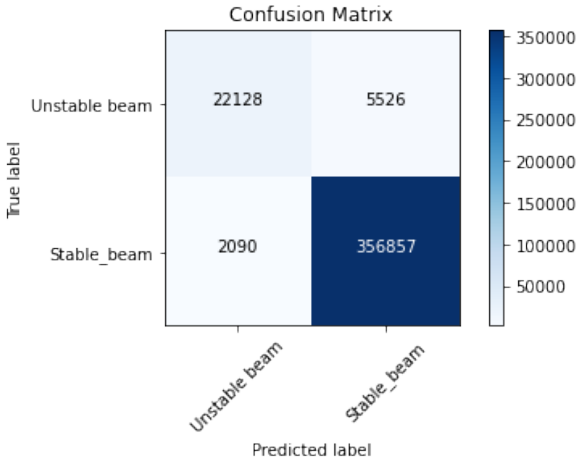


(b) Best Dilated CNN model from Bayesian hyperparameter tuning.

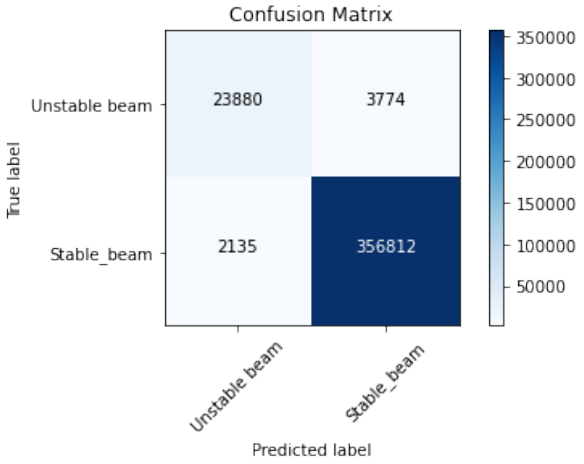


(c) Comparison of the two aforementioned Dilated CNN models with the persistence model.

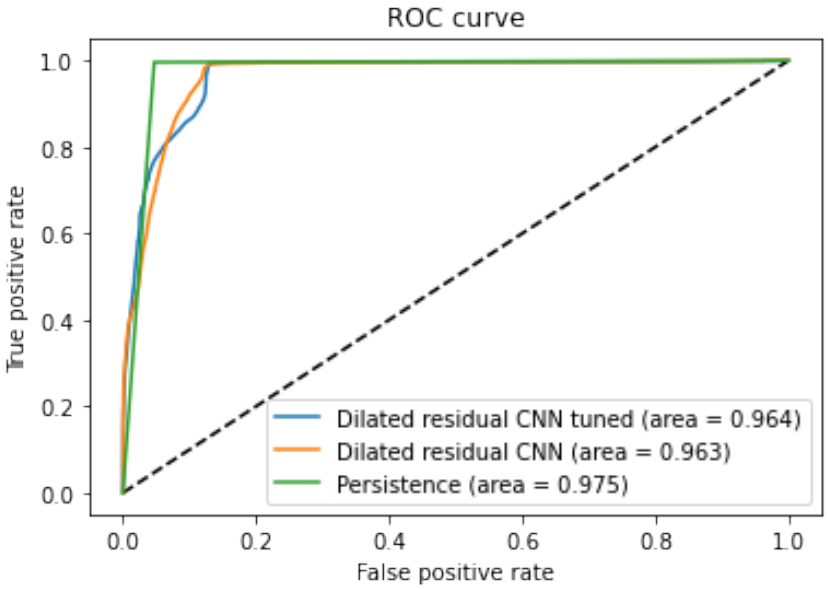
Figure 29: Dilated CNN model results for the 60 minute forecast on the test dataset.



(a) Test performance of the final Residual dilated CNN architecture.

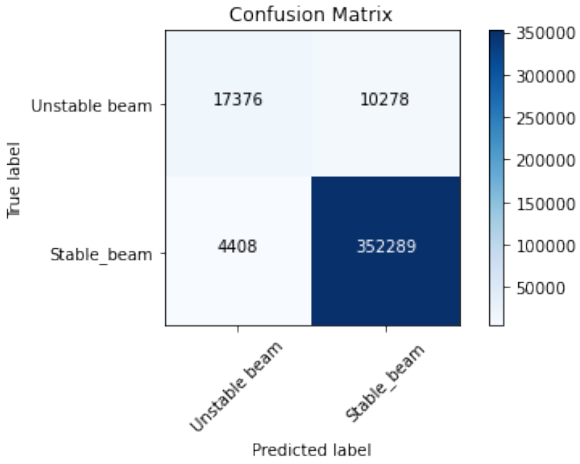


(b) Best Residual dilated CNN model from Bayesian hyperparameter tuning.

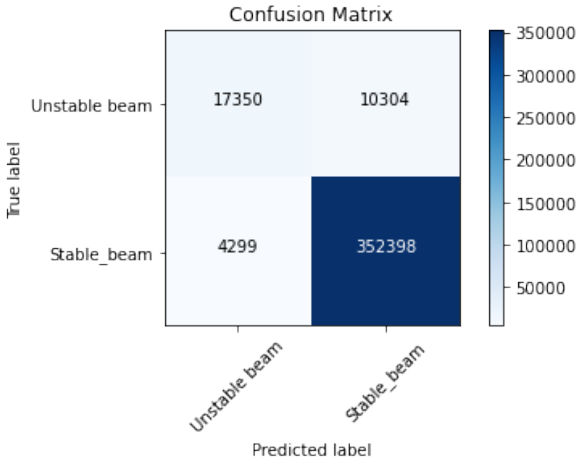


(c) Comparison of the two aforementioned Residual dilated CNN models with the persistence model.

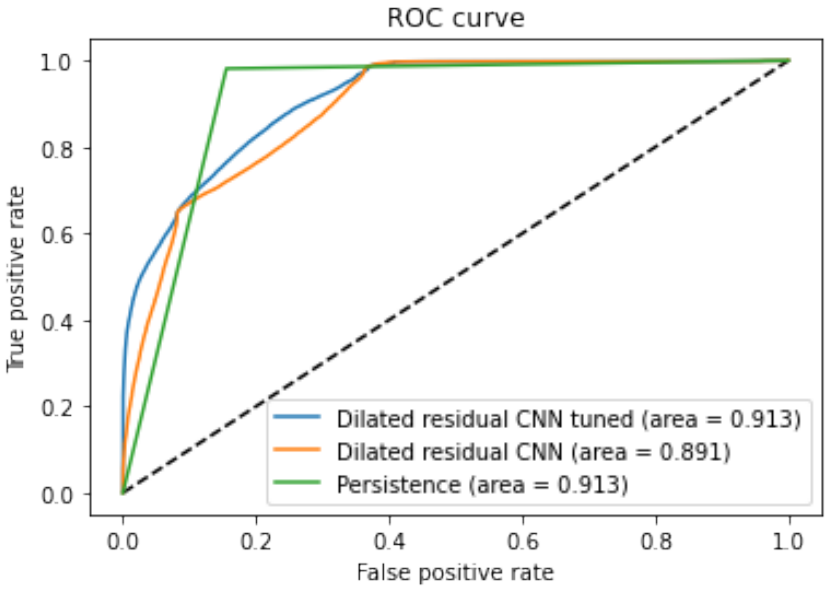
Figure 30: Residual dilated CNN model results for the 15 minute forecast on the test dataset.



(a) Test performance of the final Residual dilated CNN architecture.

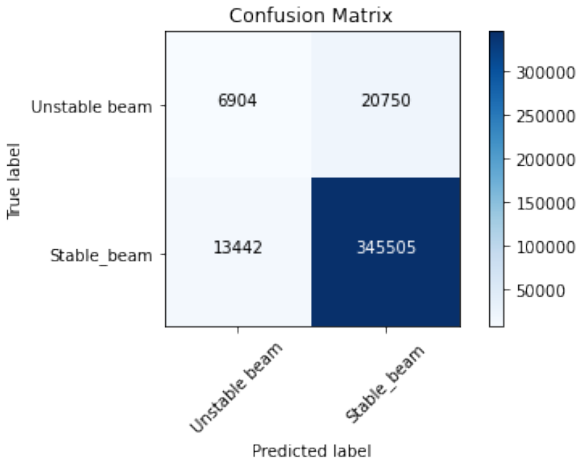


(b) Best Residual dilated CNN model from Bayesian hyperparameter tuning.

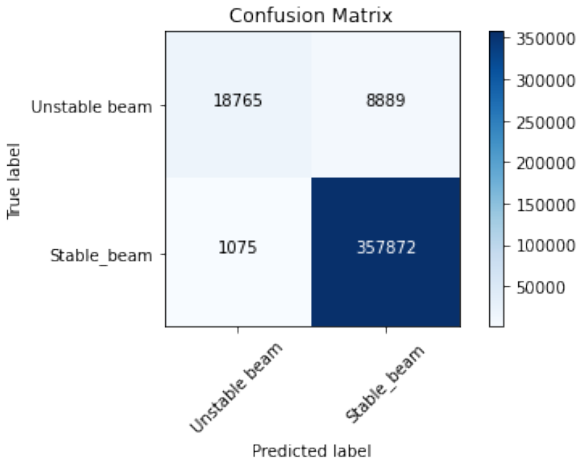


(c) Comparison of the two aforementioned Residual dilated CNN models with the persistence model.

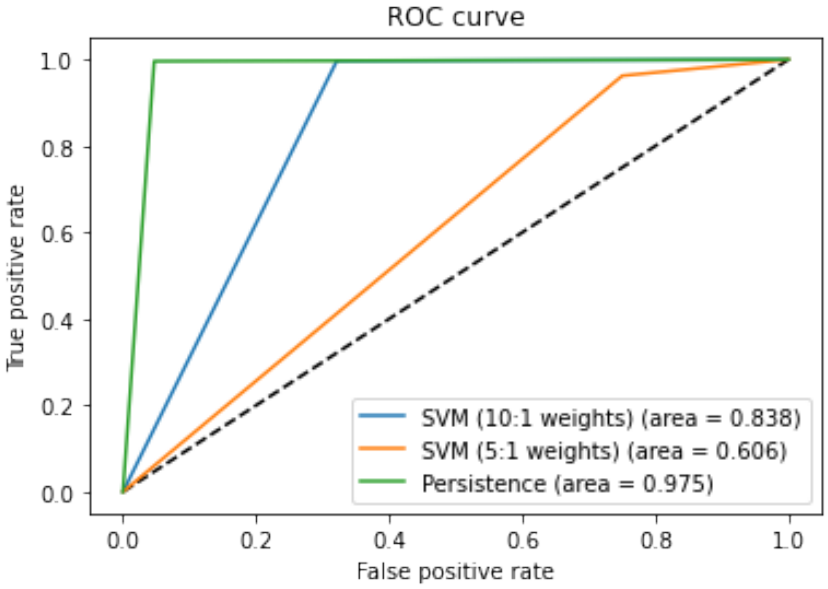
Figure 31: Residual dilated CNN model results for the 60 minute forecast on the test dataset.



(a) SVM model before grid search.

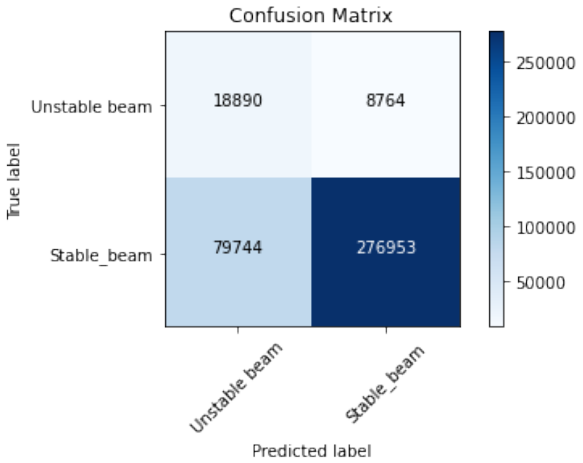


(b) Best SVM model from grid search.

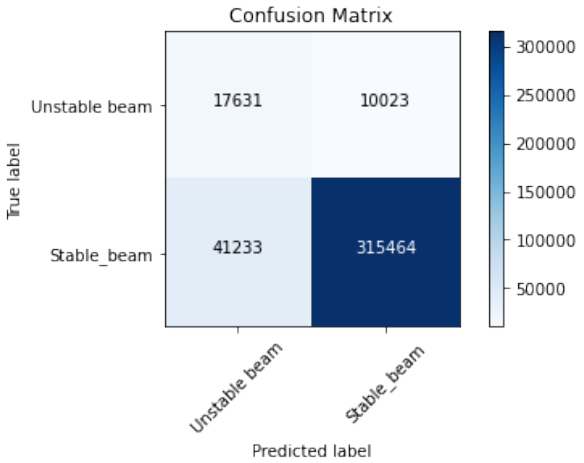


(c) Comparison of the two aforementioned SVM models with the persistence model.

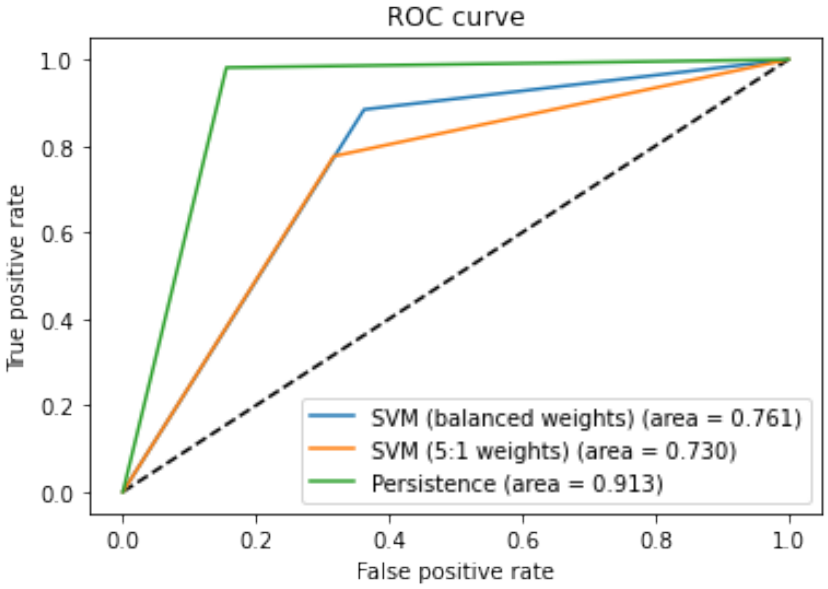
Figure 32: SVM model results for the 15 minute forecast on the test dataset.



(a) SVM model before grid search.

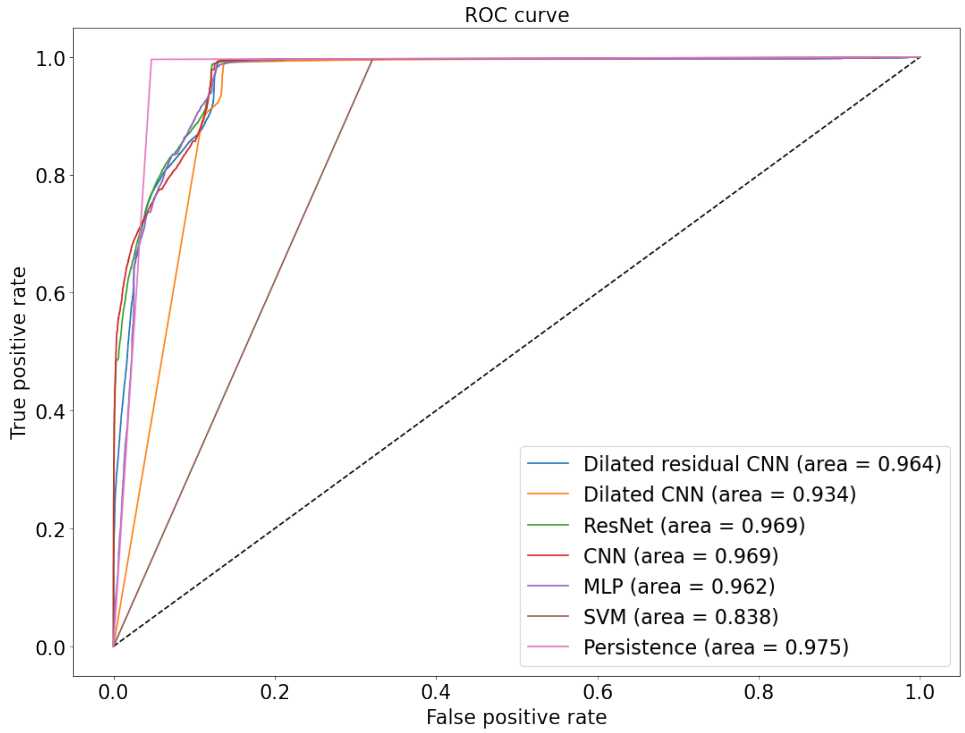


(b) Best SVM model from grid search.

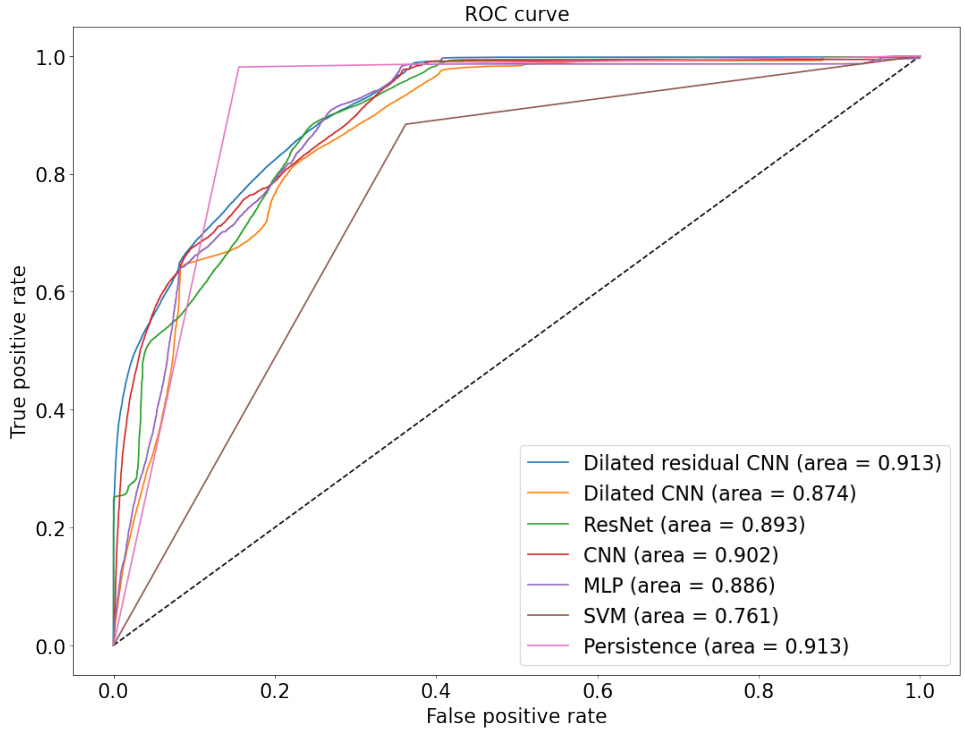


(c) Comparison of the two aforementioned SVM models with the persistence model.

Figure 33: SVM model results for the 60 minute forecast on the test dataset.



(a) Comparison of the best models with the persistence model for the 15 minute forecast.



(b) Comparison of the best models with the persistence model for the 60 minute forecast.

Figure 34: ROC curves of all models on the test dataset.

5.3 Change point detection experiments

In this section, the ability of the two change point detection algorithms, analysed in Section 4, will be compared. Specifically, they were given the same set of BCT25 data, from the 4th until the 6th of October 2021, to recognise sudden changes in the trend of BCT25 as depicted in Figure 35.

5.3.1 Direct density-ratio estimation results

For the implementation of the direct density-ratio estimation method discussed in Section 4.1, the package KLIEP, [53], has been used and modified in Python to identify change points on BCT25. First, the data from the BCT25 feature had been downsampled to 60 second bins, instead of taking into consideration all data points with frequency of 1.2 seconds. As a result, the computation time of the algorithm is 10 minutes. High voltage breakdowns could potentially give a false alarm of a beam decay to the source specialists, since the intensity of the beam current transformers, such as BCT05 and BCT25, drop to zero momentarily as in Figure 7.

Next, the values of the parameters, $k, n_{af}, n_{bf}, \mu, \sigma$ as described in Section 4.1, were tuned first on data from the 2nd until the 3rd of October. The choice of the parameters was based on trial and error experiments on this dataset. Subsequently, the model's performance was assessed on unseen data. The parameters found were the following.

k	n_{af}	n_{bf}	μ	σ
50	100	50	0.00012	10000

Table 3: The parameters used on the unseen data.

The first result from implementing direct density-ratio estimation on the BCT25 data from the 4th until the 6th of October 2021 is depicted in Figure 35a. The result indicates that the algorithm detects a change point with a delay in both datasets, whilst the parameters of the model are tuned.

Resampling enabled smoothing the noise from the data, by taking the mean in 60 second bins. Nevertheless, there is still risk for uninformative datapoints to exist. Hence, the high voltage breakdown filter, as discussed in Section 2.2.1, was applied to the test data, i.e. from the 4th until the 6th of October 2021. Next, the filtered data were fed into the KLIEP algorithm. The result of this method is illustrated in Figure 35b. The filtering method has shown an improvement compared to Figure 35a as it reduced the number of false alarms. The aforementioned idea was also implemented in the next change point detection algorithm.

5.3.2 Bayesian online change point detection method results

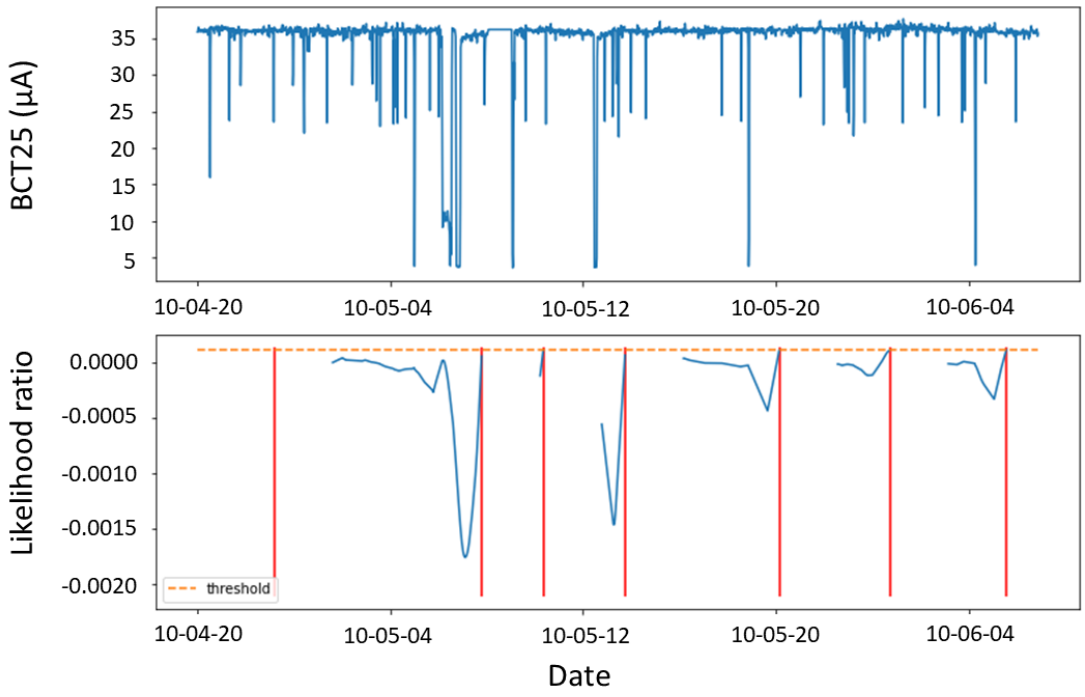
The next change point detection algorithm which was implemented to pinpoint an unanticipated decay on BCT25 data was the Bayesian online change point detection method. Its application was possible with a modified version of the Python package proposed by [71] and [55]. Similarly to the direct density-ratio estimation, the data were downsampled to 60 second bins. As discussed in Section 4.2, the parameters λ, β were first initialised to 60 and 30 datapoints respectively. This choice was based on the performance of the change point detection algorithm on the data from the 2nd until 3rd of October 2021. Subsequently, all parameters $\lambda, \alpha, \mu, \beta$ get updated according to Equation (47).

Similarly to the direct density-ratio estimation, the Bayesian online change point detection was first tested on data which have not been filtered for high voltage breakdowns. The result is provided in Figure 36a. The method recognises change points more accurately than the direct density-ratio estimation method. However, it is more prone to making a mistake, i.e. false alarm, identifying a high voltage breakdown as a change point. Given that the current method is more sensitive to high voltage breakdowns, the use of the filter is imperative. As highlighted in Figure 36b, the impact of the high voltage breakdown filter in identifying change points is substantial. What is more, the computation time of the algorithm is approximately 3 seconds.

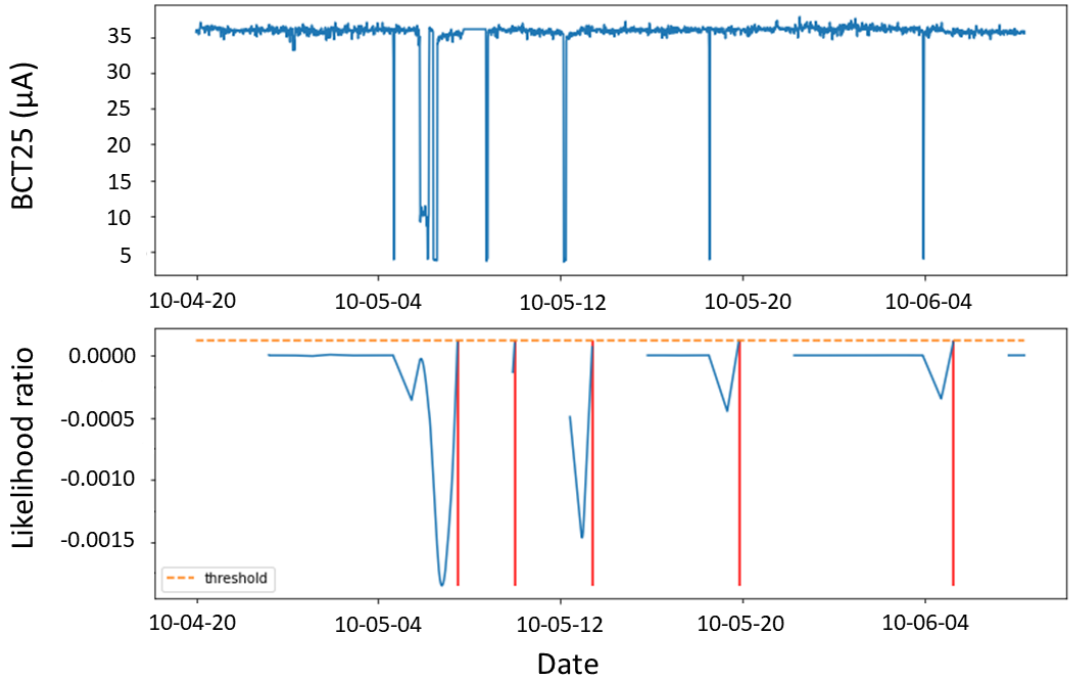
5.3.3 Change point detection experiments discussion

Taking into consideration both change point detection methods, the Bayesian online change point detection provides a faster and more accurate way of finding changes in BCT25 data. However, the method's

sensitivity to high voltage breakdowns introduces false alarms to the source specialists which are undesirable. A way to further improve the method is by upgrading the high voltage breakdown filter.

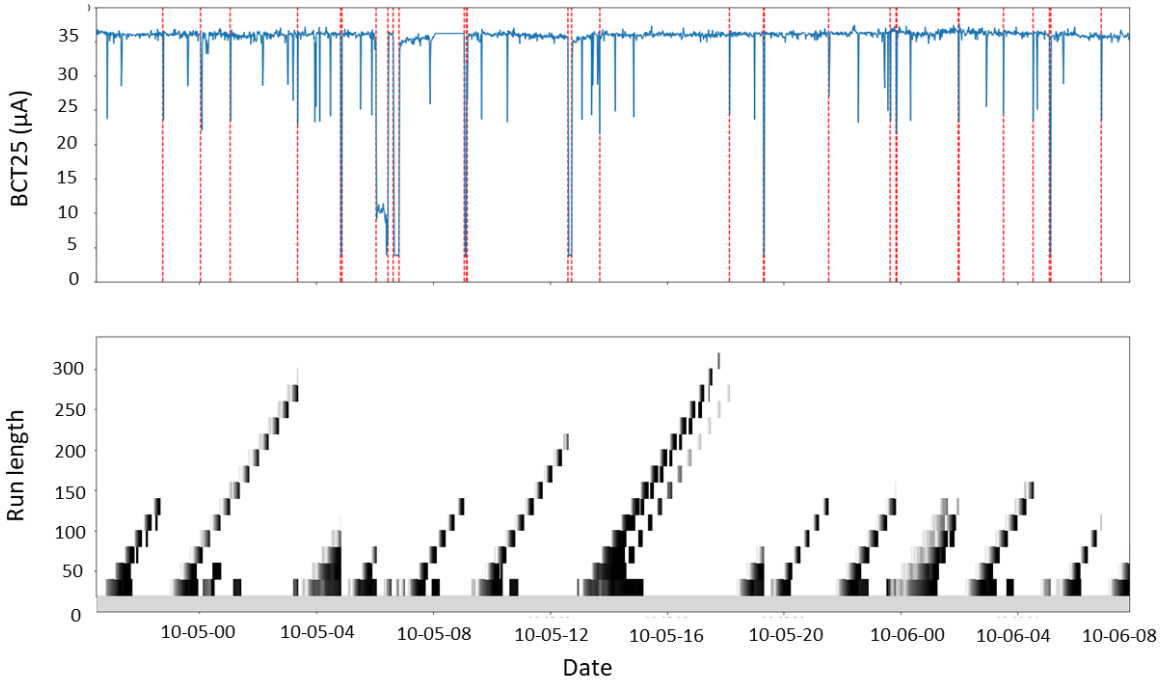


(a) Density ratio-estimation using no high voltage breakdown filter.

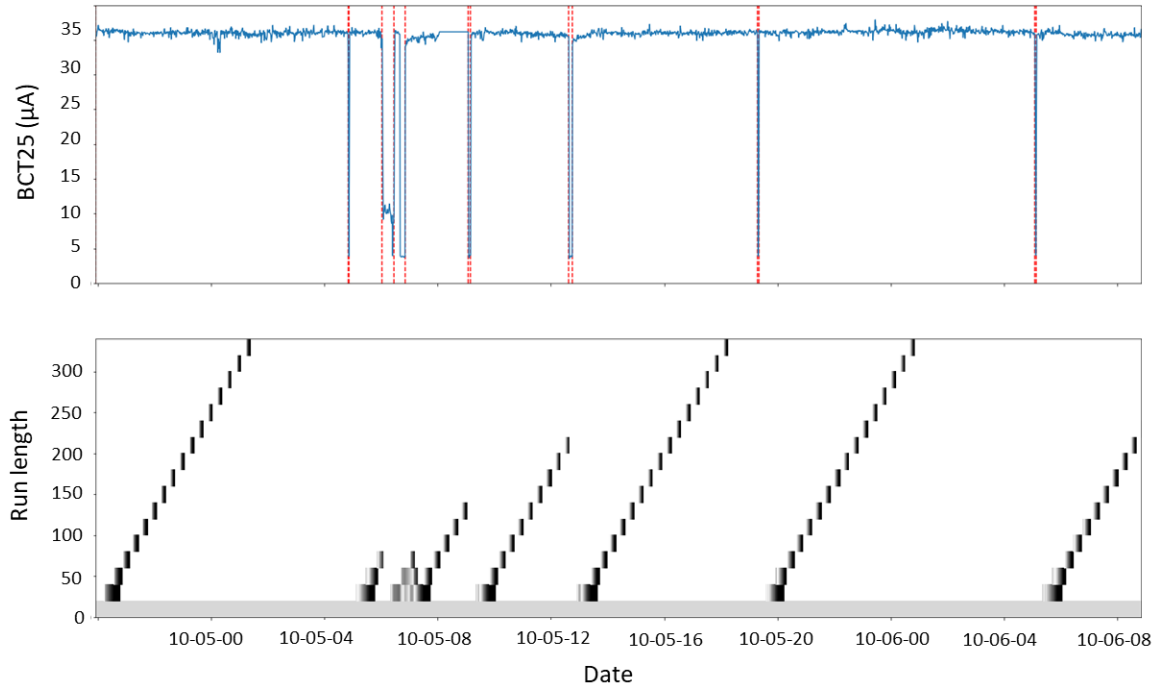


(b) Density ratio-estimation using a high voltage breakdown filter.

Figure 35: Change point detection of BCT25 for the test data using the direct density-ratio estimation method. The detected change points are displayed with red.



(a) Bayesian online change point detection without a high voltage breakdown filter.



(b) Bayesian online change point detection with a high voltage breakdown filter.

Figure 36: Change point detection of BCT25 using BOCD. The change points are highlighted with red.

6 Conclusion & Discussion

In this research, the analysis of the instability of the ion beam current at Linac3 was introduced. The primary aim of this study was to forecast a beam decay in the future, using various machine learning algorithms based on historical data from Linac3. The best performing algorithms in the 15 minute forecast, i.e. ResNet and CNN, were unable to find the necessary patterns needed to make reliable predictions on the status of the beam and outperform the baseline model, i.e. the persistence model. Moreover, in the 60 minute forecast the residual dilated CNN model reached exactly the baseline. The results outlined in Section 5.2.1 arrive to the same conclusion as mentioned in [6] and in recent research done using recurrent neural networks (LSTM) by the Beams and Industrial Control Systems group at CERN, where the goal was regression and not binary classification. Due to the multitude of ML algorithms implemented in this research and others prior to it, it can be concluded that given the current measurements from the settings and acquisitions at Linac3 it is not possible to predict a beam instability.

Further investigation inside the GTS-LHC ion source is planned to be conducted by CERN to understand in more depth its operation. Optical emission spectroscopy would be able to provide more information about the behaviour of the source [72]. This will be possible by measuring other properties of the plasma, such as the density of the lead vapor which currently are unknown. These measurements are considered critical to understand the performance of GTS-LHC. With the new measurements and features, which cannot be sufficiently inferred from any feature extraction method such as convolutional neural networks or principal component analysis, the proposed methods in the current research should be repeated for a 15 minute and a 60 minute forecast. Given that the models with the new features are able to provide significantly better predictions than the persistence model, it is desirable from the operators to implement a model for a longer-term forecast (100 minutes). For the 100 minute forecast, the dilated residual CNN model could have a sufficient amount of historic data by increasing the number of dilated layers to make longer predictions into the future. An accurate 100 minute forecast would yield more time, compared to the 15 and 60 minute forecast, to the operators to prevent a beam decay.

Despite the fact that a wide variety of ML methods has been used to predict the beam instability, there are still ML algorithms that have not yet been examined. A hybrid CNN-LSTM could use the ability of the CNN to extract features efficiently and the LSTM, which can provide a wide history from Linac3 observations, could possibly provide useful forecasts as suggested in [73]. Furthermore, the Light Gradient Boosting Machine (LightGBM) models, introduced in [74], demonstrate high accuracy and fast learning in recent time series forecasting problems and competitions [75]. A possible extension of the current experiments with SVMs would be possible with the use of the `tslearn` library in Python [76], which is tailored for time series data. Additionally, the global alignment kernel [77] is recommended for time series classification problems from `tslearn`. The aforementioned methods are recommended to be implemented after the findings of the optical emission spectroscopy.

The current research also focused on identifying change points in Linac3 data by comparing two change point detection methods. These methods are not able to provide a pre-warning to the source operator that a beam will decay. In fact, they provide a warning as soon as they take place at best. Direct density-ratio estimation proved to be robust to high voltage breakdowns. However, a notable delay between the alert of the algorithm and the actual beam decay make it difficult to be consistently implemented with little tuning of its parameters. The Bayesian online change point detection method was improved by implementing a high voltage breakdown filter. As a result, the method can observe quickly and accurately changes in the beam intensity by minimising the amount of false alarms. The ability of the change point detection method could be enhanced with the use of a better filter. Specifically, the filter has a threshold for the variability of the HT current, i.e. 0.25A for a rolling window of 40 timesteps. The filter could be more sensitive applying a lower variance threshold for a smaller time window.

A Appendix

The width of the maximum-margin hyperplane is equal to $\frac{2}{||a||}$.

Proof

Given $y_i(a^\top x_i + b) - 1 = 0$ for support vectors $x_{(+)}, x_{(-)}$ that lie on both sides of the maximum-margin hyperplane it follows

$$a^\top x_{(+)} + b = 1 \tag{49}$$
$$a^\top x_{(-)} + b = -1 \tag{50}$$

It can be seen from Figure 37 that the width of the maximum-margin hyperplane can be calculated from trigonometry

$$\text{width} = ||x_{(+)} - x_{(-)}|| \cos\theta \tag{51}$$

where θ is the angle between the normal vector a and $x_{(+)} - x_{(-)}$. Next, the dot product of $x_{(+)} - x_{(-)}$ with the unit normal $\frac{a}{||a||}$ is the following

$$(x_{(+)} - x_{(-)})^\top \frac{a}{||a||} = ||x_{(+)} - x_{(-)}|| \frac{||a||}{||a||} \cos\theta \tag{52}$$
$$= ||x_{(+)} - x_{(-)}|| \cos\theta \tag{53}$$

therefore,

$$\text{width} = (x_{(+)} - x_{(-)})^\top \frac{a}{||a||} \tag{54}$$

after substituting Equations (49, 50) in Equation (54) it follows

$$\text{width} = \frac{2}{||a||} \tag{55}$$

□

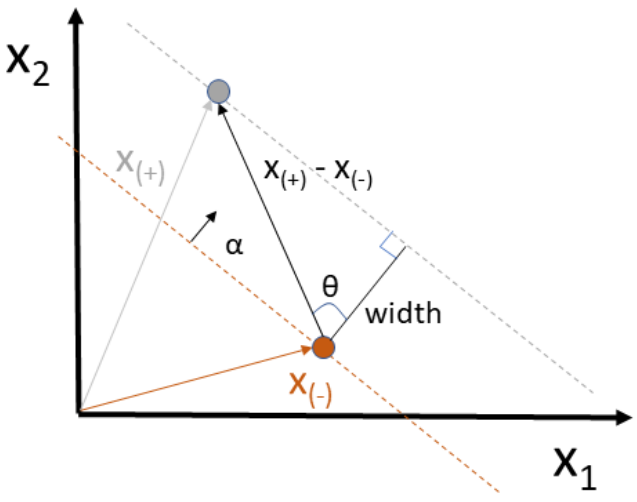


Figure 37: Example of two support vectors $x_{(+)}, x_{(-)}$ lying on the maximum-margin hyperplane. The normal of the hyperplane is denoted with a and the angle between a and $x_{(+)} - x_{(-)}$ is denoted with θ.

Standard form for optimisation problems

The notes from Boyd and Vandenberghe [25] will be followed for the formulation of continuous optimisation problems the Karush-Kuhn-Tucker conditions.

A continuous optimisation problem can be formulated in the following way:

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m, \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{aligned} \tag{56}$$

where f_0 is the objective function, f_i for $i = 1, ..., m$ are the inequality constraints and h_i for $i = 1, ..., p$ are the equality constraints.

Karush-Kuhn-Tucker (KKT) theorem

Assuming that $f_0, ..., f_m, h_1, ..., h_p$ are differentiable, the KKT theorem states that for $x^\star$ being the optimal solution for the primal problem in Equation (56), $(\lambda^\star, \nu^\star)$ being the optimal solution for the dual problem, and given that strong duality holds it follows

$$f_i(x^\star) \leq 0, \qquad i = 1, ..., m \tag{57}$$
$$h_i(x^\star) = 0, \qquad i = 1, ..., p \tag{58}$$
$$\lambda_i^\star \geq 0, \qquad i = 1, ..., m \tag{59}$$
$$\lambda_i^\star f_i(x^\star) = 0, \qquad i = 1, ..., m \tag{60}$$
$$\nabla f_0(x^\star) + \sum_{i=1}^{m} \lambda_i^\star \nabla f_i(x^\star) + \sum_{i=1}^{p} \nu_i^\star \nabla h_i(x^\star) = 0, \tag{61}$$

The optimal maximum-margin hyperplane is determined only by the support vectors and the misclassified points.

Proof

$(a^\star, b^\star, \xi^\star)$ are the optimal solutions for Equation (4) which satisfy the KKT conditions. Therefore, there is $(\lambda^\star, \mu^\star)$ for which

$$\lambda_i^\star(1 - \xi_i^\star - y_i(a^{\star\top} x_i + b^\star)) = 0 \tag{62}$$
$$\mu_i^\star \xi_i^\star = 0 \tag{63}$$
$$a^\star - \sum_{i=1}^{t} \lambda_i^\star y_i x_i = 0 \tag{64}$$
$$-\sum_{i=1}^{t} \lambda_i^\star y_i = 0 \tag{65}$$
$$C - \lambda_i^\star + \mu_i^\star = 0 \tag{66}$$

and hence $\lambda_i^\star = 0$ or $y_i(a^{\star\top} x_i + b^\star) = 1 - \xi_i^\star$ for $i = 1, ..., t$. Also, $\mu_i^\star = 0$ or $\xi_i^\star = 0$ for $i = 1, ..., t$. Using the third condition it is shown that

$$a^\star = \sum_{i=1}^{t} \lambda_i^\star y_i x_i \tag{67}$$

from the first condition it is shown that

$$y_i(a^{\star\top} x_i + b^\star)) = 1 - \xi_i^\star \tag{68}$$

for $i = 1, ..., t$ when $\lambda_i^\star \neq 0$. For Equation (68) to be true, x_i is a support vector, i.e. $\xi^\star = 0$, or x_i is misclassified, i.e. $\xi^\star > 0$. This shows that the optimal maximum-margin hyperplane relies only on the support vectors and the misclassified points.

□

Search ranges for the Bayesian hyperparameter tuning and Grid search

ReLU nodes	Dropout
$150 - 500$	$0.05 - 0.4$

Table 4: Search range for the MLP hyperparameters.

Filter	Kernel
$100 - 500$	$1 - 25$

Table 5: Search range for the CNN hyperparameters.

Filter	Kernel
$50 - 500$	$1 - 50$

Table 6: Search range for the ResNet hyperparameters.

Filter	ReLU nodes	Dropout
$150 - 350$	$150 - 350$	$0.05 - 0.4$

Table 7: Search range for the Dilated CNN hyperparameters.

Filter
$50 - 250$

Table 8: Search range for the Dilated residual CNN hyperparameters.

C	Kernel	Degree (poly)	class weight
$[0.1, 1, 10, 50, 100]$	[sigmoid, linear, poly, RBF]	$[2 - 10]$	[1:1, 5:1, 10:1, 20:1, 30:1]

Table 9: Search range for the SVM hyperparameters.

Linac3 features included in the analysis

- IP.NSRCGEN:BIASDISCAQNV: voltage of the bias disk
- IP.NSRCGEN:GASAQN: voltage of the gas feedback loop
- IP.NSRCGEN:OVEN1AQNP: power of oven 1
- IP.NSRCGEN:OVEN2AQNP: power of oven 2
- IP.SAIREM2:FORWARDPOWER: power of the Sairem 2 microwave generator
- IP.SOLINJ.ACQUISITION:CURRENT: current of the injection solenoid in the GTS-LHC
- IP.SOLCEN.ACQUISITION:CURRENT: current of the central solenoid in the GTS-LHC
- IP.SOLEXT.ACQUISITION:CURRENT: current of the extraction solenoid in the GTS-LHC
- IP.NSRCGEN:SOURCEHTAQNV: voltage of the main source high voltage
- IP.NSRCGEN:SOURCEHTAQNI: current of the main source high voltage
- ITL.BCT05:CURRENT: current of the ion beam passing the transformer ITL.BCT05
- ITF.BCT25:CURRENT: current of the ion beam passing the transformer ITF.BCT25
- ITH.BCT41:CURRENT: current of the ion beam passing the transformer ITH.BCT41

References

[1] Ludovic Dumas, Charles Hill, Denis Hitz, Detlef Küchler, Cristiano Mastrostefano, Michael O'Neill, and Richard Scrivens. *Operation of the GTS-LHC Source for the Hadron Injector at CERN*. Tech. rep. Geneva: CERN, Jan. 2007. URL: https://cds.cern.ch/record/1019180.

[2] Michael Benedikt, Paul Collier, Volker Mertens, John Poole, and Karlheinz Schindl. *LHC Design Report*. CERN Yellow Reports: Monographs. Geneva: CERN, 2004. DOI: 10.5170/CERN-2004-003-V-3. URL: https://cds.cern.ch/record/823808.

[3] *CERN's accelerator complex*. https://home.cern/science/accelerators/accelerator-complex. Online; accessed July 8, 2022.

[4] Detlef Kuchler and Max Mihailescu. "Statistical analysis of operational data of the GTS-LHC ion source". In: *Journal of Physics: Conference Series* 2244.1 (Apr. 2022), p. 012022. DOI: 10.1088/1742-6596/2244/1/012022.

[5] Alexander Hinneburg and Daniel Keim. "Optimal Grid-Clustering : Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering". In: *Proceedings of the 25 th International Conference on Very Large Databases, 1999*. 1999, pp. 506–517.

[6] Breeman Thomas, Gan Renren, Hoogendijk Lucas, Knops Pieter, Lynch Owen, Roberts Jeroen, Soons Jelle, Theodosiou Manousos, Tian Jinyao, and Warsen Nils. "Forecasting the LINAC3 Ion Beam Current". Internal note. 2021.

[7] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013.

[8] Jerome Friedman. "Stochastic Gradient Boosting". In: *Computational Statistics & Data Analysis* 38 (Feb. 2002), pp. 367–378. DOI: 10.1016/S0167-9473(01)00065-2.

[9] Samaneh Aminikhanghahi and Diane Cook. "A survey of methods for time series change point detection". In: *Knowledge and Information Systems* 51.2 (2017), pp. 339–367. DOI: 10.1007/s10115-016-0987-z.

[10] Zhiguang Wang, Weizhong Yan, and Tim Oates. "Time series classification from scratch with deep neural networks: A strong baseline". In: *International Joint Conference on Neural Networks (IJCNN)* (2017), pp. 1578–1585.

[11] Jakub Wozniak, Chris Roderick, et al. "NXCALS-Architecture and Challenges of the Next CERN Accelerator Logging Service". In: *Proceedings of International Conference on Accelerator and Large Experimental Physics Control Systems, New York, United States*. 2019, pp. 5–11.

[12] *pytimber on gitlab*. https://gitlab.cern.ch/acc-logging-team/pytimber. Online; accessed March 25, 2022.

[13] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[14] Andre Beuret, Jan Borburgh, Alfred Blas, Helmut Burkhardt, Christian Carli, Michel Chanel, A Fowler, Marine Gourber-Pace, Samantha Hancock, Michael Hourican, et al. "The LHC lead injector chain". In: *Proceedings of EPAC*. Vol. 4. Citeseer. 2004, pp. 1153–1155.

[15] Toke Kay Thomas Kövener, Detlef Küchler, and Ville Toivanen. "Lead evaporation instabilities and failure mechanisms of the micro oven at the GTS-LHC ECR ion source at CERN". In: *Review of Scientific Instruments* 91.1 (2020), p. 013320.

[16] Mohammad Kachuee, Shayan Fazeli, and Majid Sarrafzadeh. "Ecg heartbeat classification: A deep transferable representation". In: *2018 IEEE international conference on healthcare informatics (ICHI)*. IEEE. 2018, pp. 443–444.

[17] Gian Antonio Susto, Angelo Cenedese, and Matteo Terzi. "Time-series classification methods: Review and applications to power systems data". In: *Big data application in power systems* (2018), pp. 179–220.

[18] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. "Deep learning for time series classification: a review". In: *Data mining and knowledge discovery* 33.4 (2019), pp. 917–963.

[19] John Cristian Borges Gamboa. "Deep Learning for Time-Series Analysis". In: *CoRR* abs/1701.01887 (2017). URL: http://arxiv.org/abs/1701.01887.

[20] Anastasia Borovykh, Sander Bohte, and Cornelis Oosterlee. "Dilated Convolutional Neural Networks for Time Series Forecasting". In: *Journal of Computational Finance* (Mar. 2017). DOI: 10.21314/JCF.2019.358.

[21] Mahmoud Okasha. "Using Support Vector Machines in Financial Time Series Forecasting". In: *International Journal of Statistics and Applications* 4 (Feb. 2014), pp. 28–39.

[22] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine Learning* 20.3 (1995), pp. 273–297. DOI: 10.1007/BF00994018.

[23] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. USA: Cambridge University Press, 2014.

[24] Vladimir Naumovich Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[25] Stephen Boyd and Lieven Vandenberghe. *Convex Optimisation*. 2004, p. 727. URL: https://www.cambridge.org/9780521833783.

[26] Li Feng. *Shandong University Lecture 6 - Support Vector Machine*. https://funglee.github.io/ml/slides/Lecture6-SVM.pdf. Online; accessed May 10, 2022.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[28] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[29] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[30] Springer Verlag GmbH, European Mathematical Society. *Encyclopedia of Mathematics*. Website. URL: https://encyclopediaofmath.org/wiki/Borel_function. Online; accessed June 10, 2022.

[31] Ah Chung Tsoi. "Gradient based learning methods". In: *International School on Neural Networks, Initiated by IIASS and EMFCSC* (1997), pp. 27–62.

[32] David Rumelhart, Geoffrey Hinton, and Ronald Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0.

[33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[34] Deepika Jaswal, Sowmya Vishvanathan, and Soman Kp. "Image Classification Using Convolutional Neural Networks". In: *International Journal of Scientific and Engineering Research* 5 (June 2014), pp. 1661–1668. DOI: 10.14299/ijser.2014.06.002.

[35] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 448–456. URL: https://proceedings.mlr.press/v37/ioffe15.html.

[36] Min Lin, Qiang Chen, and Shuicheng Yan. "Network In Network". In: *CoRR* abs/1312.4400 (2014).

[37] Yahui Chen. "Convolutional neural network for sentence classification". MA thesis. University of Waterloo, 2015.

[38] Batzner, Kilian. *Convolutions in Autoregressive Neural Networks*. https://theblog.github.io/post/convolution-in-autoregressive-neural-networks. Online; accessed May 10, 2022. 2019.

[39] Hans Weytjens and Jochen De Weerdt. "Process outcome prediction: CNN vs. LSTM (with attention)". In: *International Conference on Business Process Management*. Springer. 2020, pp. 321–333.

[40] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[41] José Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez, and Alicia Troncoso. "Deep Learning for Time Series Forecasting: A Survey". In: *Big Data* 9 (Dec. 2020). DOI: 10.1089/big.2020.0159.

[42] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. "Understanding the effective receptive field in deep convolutional neural networks". In: *Advances in neural information processing systems* 29 (2016).

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[44] Danilo Piparo, Enric Tejedor, Pere Mato, Luca Mascetti, Jakub Moscicki, and Massimo Lamanna. "SWAN: A service for interactive analysis in the cloud". In: *Future Generation Computer Systems* 78 (2018), pp. 1071–1078. DOI: https://doi.org/10.1016/j.future.2016.11.035.

[45] Marc Lavielle and Gilles Teyssiere. "Adaptive detection of multiple change-points in asset price volatility". In: *Long memory in economics*. Springer, 2007, pp. 129–156.

[46] Anton Kondratev, Katri Salminen, Jussi Rantala, Timo Salpavaara, Jarmo Verho, Veikko Surakka, Jukka Lekkala, Antti Vehkaoja, and Philipp Müller. "A comparison of online methods for change point detection in ion-mobility spectrometry data". In: *Array* 14 (2022), p. 100151. DOI: https://doi.org/10.1016/j.array.2022.100151.

[47] Yoshinobu Kawahara and Masashi Sugiyama. "Sequential Change-Point Detection Based on Direct Density-Ratio Estimation". In: *Statistical Analysis and Data Mining* 5.2 (Apr. 2012), pp. 114–127. DOI: 10.1002/sam.10124.

[48] Michèle Basseville and Igor Nikiforov. *Detection of Abrupt Change Theory and Application*. Vol. 15. Apr. 1993.

[49] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. "Change-point detection in time-series data by relative density-ratio estimation". In: *Neural Networks* 43 (July 2013), pp. 72–83. DOI: 10.1016/j.neunet.2013.01.012.

[50] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Buenau, and Motoaki Kawanabe. "Direct Importance Estimation with Model Selection and Its Application to Covariate Shift Adaptation". In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt, D. Koller, Y. Singer, and S. Roweis. Vol. 20. Curran Associates, Inc., 2008. URL: https://proceedings.neurips.cc/paper/2007/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper.pdf.

[51] Yoshinobu Kawahara, Takehisa Yairi, and Kazuo Machida. "Change-point detection in time-series data based on subspace identification". English. In: *Proceedings of the 7th IEEE International Conference on Data Mining, ICDM 2007*. Proceedings - IEEE International Conference on Data Mining, ICDM. 7th IEEE International Conference on Data Mining, ICDM 2007 ; Conference date: 28-10-2007 Through 31-10-2007. Dec. 2007, pp. 559–564. DOI: 10.1109/ICDM.2007.78.

[52] F. Desobry, M. Davy, and C. Doncarli. "An online kernel change detection algorithm". In: *IEEE Transactions on Signal Processing* 53.8 (2005), pp. 2961–2974. DOI: 10.1109/TSP.2005.851098.

[53] Johannes Kulick and David Tolpin. *KLIEP*. https://github.com/kminoda/kliep. Online; accessed February 18, 2022.

[54] Ryan Adams and David MacKay. *Bayesian Online Changepoint Detection*. 2007. DOI: 10.48550/ARXIV.0710.3742.

[55] Max Mihailescu Detlef Kuchler. "Ideas for the Linac3 Source ML Analysis". Internal note. 2020.

[56] Gregory Gundersen. *Bayesian Online Changepoint Detection*. http://gregorygundersen.com/blog/2019/08/13/bocd. Online; accessed February 23, 2022.

[57] Daniel Fink. *A Compendium of Conjugate Priors*. https://courses.physics.ucsd.edu/2018/Fall/physics210b/REFERENCES/conjugate_priors.pdf. Online; accessed February 4, 2022. 1997.

[58] Kevin Murphy. *Conjugate Bayesian analysis of the gaussian distribution (Tech. Rep.)* https://www.cs.ubc.ca/~murphyk/Papers/bayesGauss.pdf. Online; accessed March 5, 2022. 2007.

[59] Paul Fearnhead and Zhen Liu. "On-line inference for multiple changepoint problems". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69.4 (2007), pp. 589–605.

[60] Jeremias Knoblauch and Theodoros Damoulas. "Spatio-temporal Bayesian on-line changepoint detection with model selection". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2718–2727.

[61] Kai Ming Ting. "Confusion Matrix". In: *Encyclopedia of Machine Learning and Data Mining*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2017, pp. 260–260. DOI: 10.1007/978-1-4899-7687-1_50.

[62] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. 2016. DOI: 10.48550/ARXIV.1603.04467.

[63] François Chollet et al. *Keras*. https://keras.io. Online; accessed March 3, 2022.

[64] Giuseppe Bonaccorso. *Machine learning algorithms*. Packt Publishing Ltd, 2017.

[65] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[66] Diederik Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980.

[67] Eric Brochu, Vlad M Cora, and Nando De Freitas. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning". In: *arXiv:1012.2599* (2010).

[68] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems* 25 (2012).

[69] Fernando Nogueira. *Bayesian Optimization: Open source constrained global optimization tool for Python*. https://github.com/fmfn/BayesianOptimization. Online; accessed March 3, 2022.

[70] Steven LaValle, Michael Branicky, and Stephen Lindemann. "On the relationship between classical grid search and probabilistic roadmaps". In: *The International Journal of Robotics Research* 23.7-8 (2004), pp. 673–692.

[71] Johannes Kulick. *Bayesian Changepoint Detection Code*. https://github.com/hildensia/bayesian_changepoint_detection. Online; accessed March 3, 2022.

[72] Kronholm Risto, Kalvas Taneli, Koivisto Hannu, Kosonen Sami, Marttinen Miha, Neben Derek, Muneer Sakildien, Olli Tarvainen, and Ville Toivanen. "ECRIS plasma spectroscopy with a high resolution spectrometer." In: *The Review of scientific instruments* 91 1 (2020), p. 013318.

[73] Zuhaira M. Zain, Nazik M. Alturki, and Mohamed Hamdy. "COVID-19 Pandemic Forecasting Using CNN-LSTM: A Hybrid Approach". In: *Journal of Control Science and Engineering* 2021 (Jan. 2021). DOI: 10.1155/2021/8785636.

[74] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 3149–3157.

[75] Yuyi Zhang, Ruimin Ma, Jing Liu, Xiuxiu Liu, Ovanes Petrosian, and Kirill Krinkin. "Comparison and Explanation of Forecasting Algorithms for Energy Time Series". In: *Mathematics* 9.21 (2021). DOI: 10.3390/math9212794.

[76] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. "Tslearn, A Machine Learning Toolkit for Time Series Data". In: *Journal of Machine Learning Research* 21.118 (2020), pp. 1–6. URL: http://jmlr.org/papers/v21/20-091.html.

[77] Marco Cuturi. "Fast global alignment kernels". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 929–936.