Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Performance of different machine learning techniques for forecasting of particle accelerator interlocks

Master Thesis

in High Energy Physics

Department of Physics

ETH Zurich

written by

Mélissa Zacharias

supervised by

Dr. A. Adelmann (ETH)

scientific advisers

Sichen Li (ETH)
Dr. Jochem Snuverink (PSI)
Jaime Coello (PSI)
Dr. Perez Cruz Fernando (SDSC)

May 20, 2020

Abstract

Two machine learning algorithms were applied to decrease beam time loss in the High Intensity Proton Accelerator complex (HIPA) by forecasting interlock events. The random forest and a convolutional neural network (CNN) trained on recurrence plots constructed from multivariate time series were trained on HIPA data from September to December 2019. The random forest model reached a beam time loss of 19.4 ± 0.5 seconds per interlock, compared to the baseline loss for no intervention of 25 seconds per interlock. Application of the model thus has the potential to save beam time. The CNN reached a beam time loss of 20.5 ± 0.7 seconds per interlock. Preliminary testing on live predictions using the random forest model didn't result in timely interlock predictions. Simulated live testing on the CNN indicated possible data leakage from the interlock time stamps labeling. Measures have been taken to eliminate this leakage and subsequent tests of the CNN model are promising but in need of further tuning. The number of relevant features used in both models could be reduced from the original 311 features to about 50 for the Random Forest model and 11 for the CNN model using various feature selection methods. Correlation analysis of the selected features in the different sets suggests relations that may warrant further investigation.

Contents

Acknowledgement

I would like to thank Dr. A. Adelmann for the opportunity to do a master thesis in his group as well as for his assistance and guidance during this project. My thanks are also extended to Sichen Li as well as Dr. Jochem Snuverink and Jaime Coello for their expertise and very pleasant collaboration. Expertise provided by Dr. Perez Cruz Fernando and Anastasia Pentina was also greatly appreciated.

I would like to offer my thanks to Renato Belotti and Arnau Albà Jacas for the friendly office atmosphere and occasional coding advice. Further thanks are extended to the participants of the ML Luncheon for their useful critique and comments as well as to all the members of the Accelerator Modelling and Advanced Simulations Group for providing such an enjoyable working environment.

Lastly I would like to thank Andreas Müller for his support and writing advice particularly during the last stages of this thesis.

Chapter 1

Introduction

1.1 The Problem setup

The aim of this project is to develop a model that is able to predict interlocks for the High Intensity Proton Accelerator complex (HIPA) at PSI. HIPA generates a continuous proton beam of 1.2 MW, 2 mA and 590 MeV and is used for a large variety of experiments ranging from solid state physics to biology and medicine. Figure (7.1), included in the appendix, gives an overview of the HIPA infrastructure as well as the locations of different sensors and monitors, subsequently called channels, some of which will be used as features in this project.

The interlock system is a safety measure of the HIPA particle accelerator. Thresholds have been defined for the values of certain channels and if these thresholds are crossed the system shuts down the beam operation. This interruption in beam operation is what we call an interlock, see figure (1.1) .

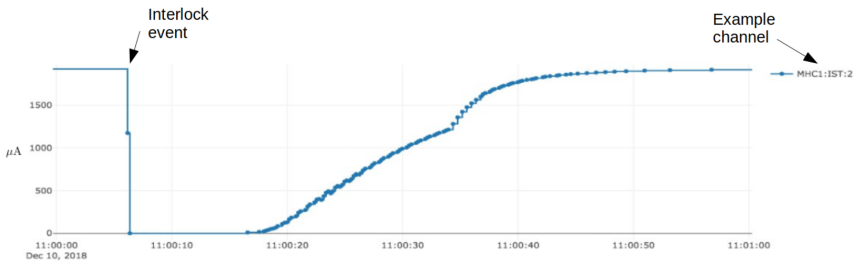


Figure 1.1: Example of an interlock event observed in the channel MHC1:IST:2

Being able to predict and avoid interlocks would allow us to save a significant amount of beam time as each interlock results in about 25 seconds of beam time loss, as illustrated in figure (1.2). Correctly predicting an interlock allows the operators to prevent it by reducing the beam intensity.

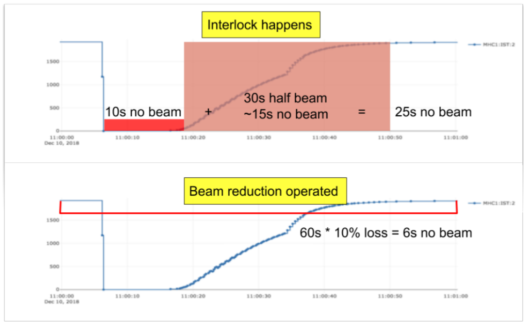


Figure 1.2: Beam time loss in the case of an interlock (above) and in the case of a prevented interlock (below). Figure courtesy of Li Sichen

We assume that a beam intensity reduction of 10% is sufficient to prevent the interlock. The reduction in beam intensity then corresponds to about 6 seconds of beam time loss. By correctly predicting an interlock, which results in 25 seconds of beam time loss, we can thus save about 19 seconds of beam time. As interlocks are responsible for about 20% of the total beam time loss in HIPA this would be a significant improvement.

The interlocks are categorized by type and location. Each interlock has at least one type and one location, though multiple types can be attributed to one interlock if no single category could be identified.

Tables 1.1 and 1.2 show the possible types and origin locations of the interlocks in the considered dataset as of the time of writing.

Type	**Description**
Electrostatic	Interlock related to electrostatic elements
Transmission	Interlock related to transmission through target
Losses	Interlock related to beam losses
Other type	Interlock related to another type

Table 1.1: The different types of interlocks is the dataset. List composed by Jochem Snuverink and Jaime Maria Coello de Portugal Martinez Vazquez.

Location	**Description**
Ring	Interlock related to the Ring cyclotron
P-Channel	Interlock related to the P-Channel (incl. SINQ)
Other Location	Interlock related to another location
Operator	Interlock related to an operator
Unknown	Unknown

Table 1.2: The possible origin locations of Interlocks. List composed by Jochem Snuverink and Jaime Maria Coello de Portugal Martinez Vazquez.

The preliminary selection of channels to include in the dataset has been done by Jochem Snuverink and

David Reggiani. Some channels are interlock channels, which means a threshold value has been defined for them above which an interlock is triggered. These are included. Related channels, identified by expert opinion, are also included as well as others that are deemed relevant from a physics perspective.

Formulation as classification problem The approach chosen here is to reformulate the time series predictions into a classification problem. Windows of multivariate time series were defined and classified into the binary categories “interlock”and “not an interlock”. The second category will henceforth be referred to as “stable”for simplicity. To obtain a window, the time series from each of the chosen channels is divided into parts of a specified length. The length of the windows may vary and is one of the parameters of the considered models. Each window is then given a label, namely “interlock”or “stable”, see figure (1.3).

An interlock window is defined as a window whose last time stamp is at 5 seconds before the occurrence of an interlock. In the regions between interlocks we discard the first and last 10 minutes of that region. The stable windows are then the consecutive, non-overlapping windows in the remaining part of the region between interlocks, see figure (1.3).

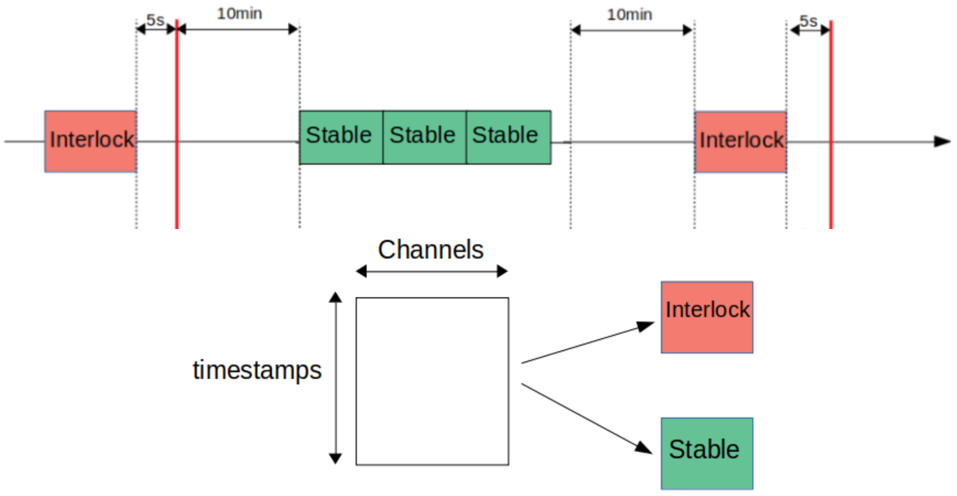


Figure 1.3: Above: Illustration of the window cutting process. The “interlock” windows are cut 5s before the interlock event, represented here by the red lines. The “stable” windows are cut in between buffers of 10 minutes to the closest interlock event of window. Below: Illustration of a window. The windows have the dimensions (number of timestamps, number of channels) and are labeled either “interlock” or “stable”.

The buffer of 5 seconds between the end of the interlock window and the actual event was chosen to give operators a chance to react to the prediction as well as to account for lag in the data recording. There should be enough time for the prediction to be useful meaning the operators should be able to act on it, assess the situation and reduce the beam intensity accordingly. One also has to account for lag in the recording of the channels. The interlock may not have happened exactly at the time stamp it was allocated to and the recording of the channels may not have been completely synchronized. The 5 seconds buffer guarantees that none of the channels in the interlock window contain time steps that would actually correspond to the interlock event itself.

The buffer of 10 minutes between the interlock event and the first stable window was chosen to make sure that the stable windows are no longer influenced by signals correlated to the interlock.

At the time of writing changes have been made to the allocation of the interlock time stamp. The details and reasoning behind this change are described in chapter 5.

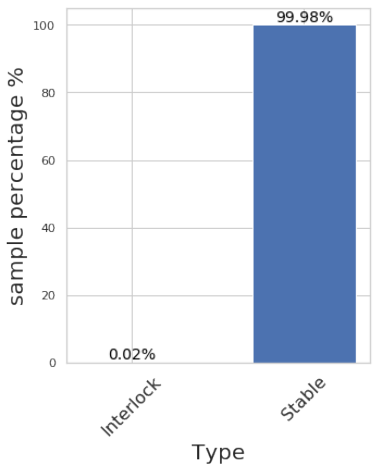
1.2 A look at the data

The data used in this project has been collected from the 19.09.2019 to the 21.12.2019.

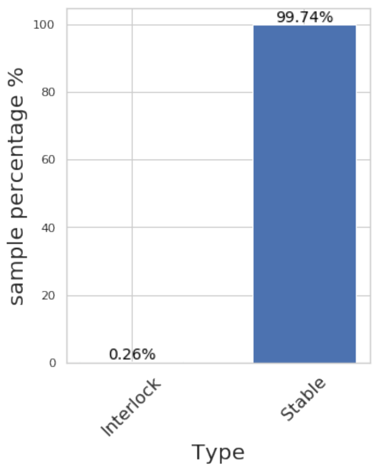
In order to simplify the data collection and cleanup process and to reduce dependency on the HIPA archiver, Jaime Coello developed a separate file storage.

The data is collected from the HIPA archiver and all chosen channels are merged into one pandas dataframe for each day. The channel data is then interpolated and the interlocks are categorized into types and locations detailed in tables (1.1) and (1.2). All beam development days have been omitted during loading of the dataframe.

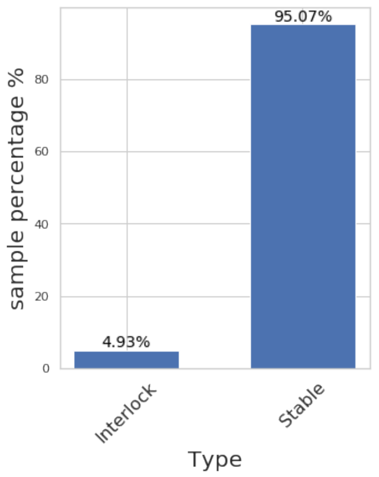
In this work different window sizes, see figure (1.3), have been used which has a relevant effect on the class distributions. The window sizes are given in the amount of data points, time stamps, that they contain. As the data is sampled at a frequency of 5 Hz one time stamp corresponds to 0.2 s. Figure (1.4) presents the class distributions for windows of length 1 time stamp, 25 time stamps (5 s), 500 time stamps (100 s) and 2000 time stamps (400 s).



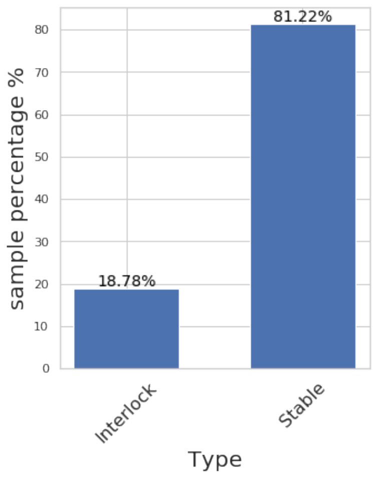
a 1 time stamp



b 25 time stamps



c 500 time stamps



d 2000 time stamps

Figure 1.4: Distribution of interlock windows and stable windows in the 2019 data for different window lengths.

The dataset is highly unbalanced for all considered window sizes. This needs to be taken into account in the training and evaluation of the considered models.

1.2.1 Interlock distributions

The number and distribution of interlock events, as well as the distribution of their categories, seem to be fairly consisted over the considered dataset. Note that the gaps along the x-axis of the following figures correspond to the beam development days, that have been omitted from the dataset. Figure (1.5) shows the number of recorded interlocks per day.

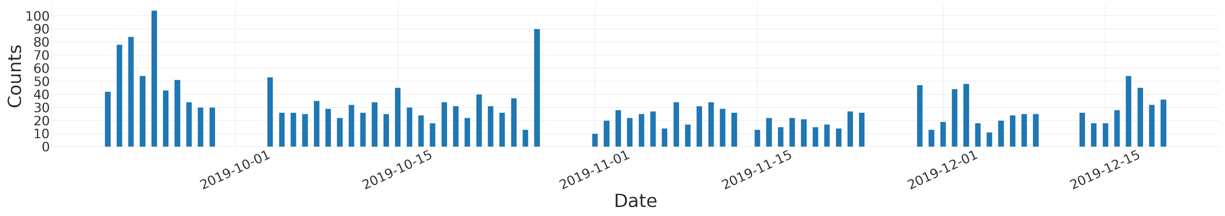


Figure 1.5: Distribution of the number of interlocks per day.

One notices that the regions with the highest interlock counts is between September 20th and September 26th. The interlock count generally decreases until November 29th with one outlier at October 27th, and seems to increase again towards the end of the measurement.

Figure (1.6) presents the distribution of interlock types per day in the dataset.

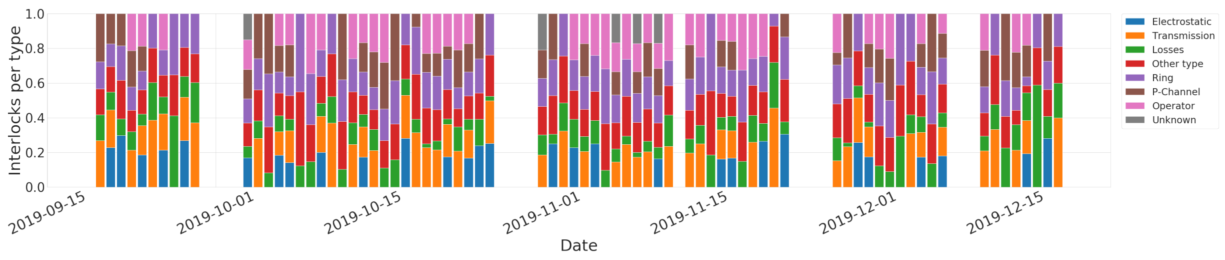


Figure 1.6: Distribution of the types of interlock events per day.

The category “Unknown” is the least represented, present in only five days in the dataset. The categories “Other type”, “Losses” and “Ring” are present in every day of the dataset, while the absent on some days in varying proportions. The distributions seem to be fairly even while the higher and lower interlock numbers discussed above are also apparent in this representation.

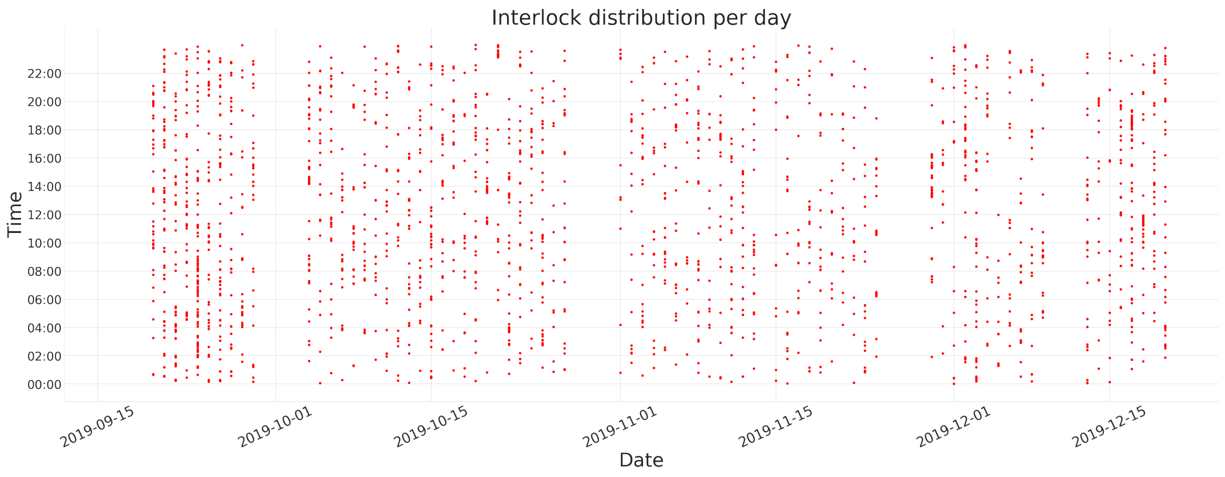


Figure 1.7: Occurrence of interlock events per day.

Figure (1.8) shows the distribution of interlock events by type and location in the dataset.

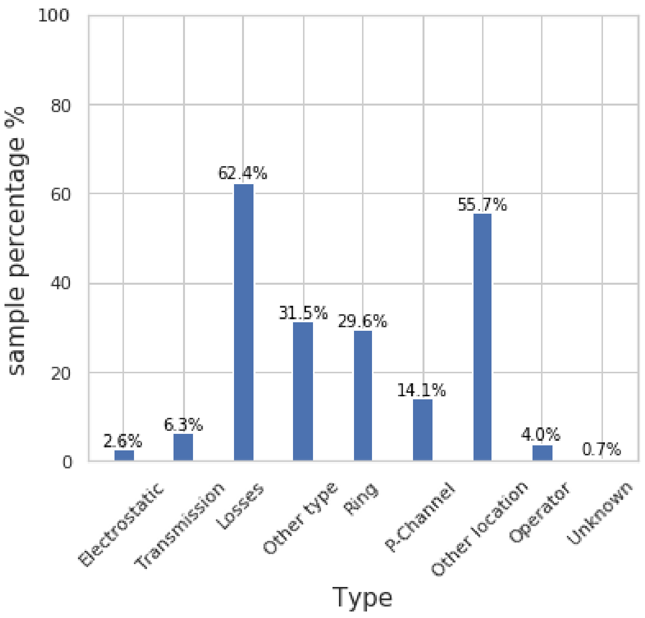


Figure 1.8: Overall distribution of Interlock events by type and location. The type explanations can be found in tables (1.1) and (1.2)

Note that one sample can be assigned to more than one category. The categories “Losses” and “Other location” are most frequent in this dataset, “Electrostatic” and “Unknown” the rarest. As the overall number of samples per category is vastly different, the classification performance of a model should also be investigated per interlock category.

1.3 Evaluation Metrics

For an imbalanced dataset such as the present one, evaluation of the model performance through the classification accuracy is not sufficient. Receiver operating characteristic plots, as well as a target defined to minimize beam time loss, will serve as the main tools to characterize the performance of each model.

Receiver operating characteristic (ROC) plots ROC curves are tools for visualizing and evaluation binary classification models. They show the true positive rate (TPR) against the false positive rate (FPR) of the model predictions as a function of the discrimination threshold [1].

True positive rate:

$$TPR = \frac{TP}{TP + FN}$$

False positive rate:

$$FPR = \frac{FP}{FP + TN}$$

To produce a ROC plot, the classifier is made not to output a label, but a score of “closeness to the positive class” between 0 and 1. This score is then evaluated at a range of threshold values. If the score is above the threshold, the sample is labeled as a member of the positive class.

Figure (1.9) illustrates the interpretation of a ROC plot.

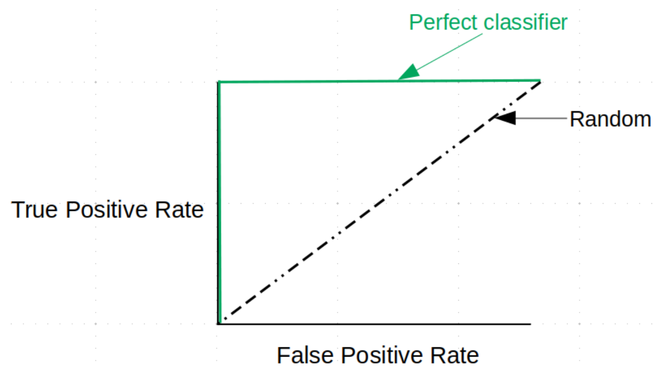


Figure 1.9: Illustration of how to interpret a ROC plot.

The diagonal line represents a classifier that would randomly guess the class label of each sample. Random guesses always yield the same amount of FP as TP and thus produce points on the ROC curve from (0.0,0.0) to (1.,1.) which gives us the diagonal. A perfect classifier would generate a TPR of 1. no matter the threshold and thus result in the green line, shown in Figure (1.9).

One of the advantages of using ROC plots, is that they are invariant to changes in the class distribution. As our interlock/stable distributions do depend on the chosen window size, this property allows the comparison of models using different window sizes.

Related to the ROC plots is the Area Under the ROC Curve (AUC) metric. The AUC gives the probability that the evaluated model will give a random positive sample a higher value than a random negative sample. It can take values in (0,. . . ,1) where 0.5 represents randomness.

$$AUC = \int_{x=0}^{1} TPR(FPR(x)^{-1})dx \tag{1.1}$$

Target definition The considerations illustrated in figure (1.2) allow us to define a target for the minimal performance of a model necessary to save beam time. We assume in the following that every correctly predicted interlock is prevented, thus that the success rate of the prevention by current reduction is 100%. Then every TP saves 19 seconds of beam time. Every FP causes 6 seconds of beam time loss. As interlocks lead to 1%-2% beam time loss of 90% beam availability, a weight of 45 is assigned to the false positives. This leads us to:

$$\begin{aligned} \text{target} &= \max(19TPR - 6 * 45FPR) \\ &= \max(TPR - 14.2FPR) \\ &\approx \max(TPR - 10FPR) \end{aligned} \tag{1.2}$$

In order to save beam time through the usage of an interlock predicting model, the model needs to reach a target value above zero. A target value of zero would mean that no beam time would be saved through employment of the model, a value below zero suggests that beam time loss would increase with the use of the model. Thus only models presenting a positive target value would be useful in practice.

Figure (1.10) shows how to interpret this target on a ROC plot.

Beam time lost Another way to assess model performance, used here for the final comparisons, is to calculate the beam time lost in seconds per interlock. This metric is based upon the same considerations as detailed above.

$$\text{Beam time lost} = (1 - TPR) * 25 + TPR * 6 + FPR * 45 * 6 \tag{1.3}$$

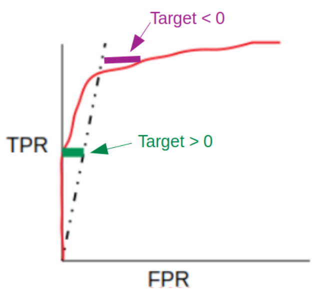


Figure 1.10: Illustration of the beam time loss target on a ROC plot.

With no intervention one obtains a baseline loss of 25 s per interlock. In case of a correct prediction (TP) 6 s are lost, in case of a wrongly predicted interlock (FP) 6 s are lost. The beam time lost can thus present values between 6 s and 25 s with 6 s being the ideal case.

Chapter 2

Theory

2.1 Random Forest

The following is intended to be a short summary of the random forest classification model, specifically of the variation used in this project. For a comprehensive discussion the reader is referred to [2].

A random forest is an ensemble of decision trees. Each of these trees is built using a subset of the training samples and features drawn with replacement from the respective sets.

Figure (2.1) presents a toy example of such a decision tree.

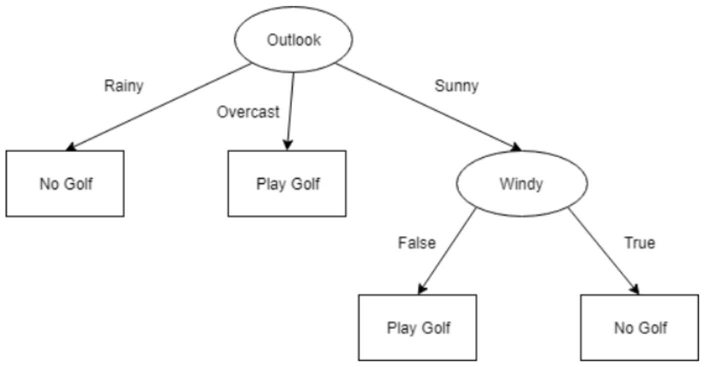


Figure 2.1: Toy example of a decision tree [3].

A decision tree consists of nodes, branches and leafs. The nodes each contain a feature and a question for the value of that feature. The node content is learned during training. The branches represent the answers to the question in the origin node. The leafs, nodes at the end of the tree, contain class labels. A new sample given to the tree is classified by moving from node to node according to how the values of its features “answer” the questions in the nodes, until it reaches a leaf. In the toy example days are classified into the binary categories golf/no golf. Consider the sample (Outlook = Sunny, Windy = False). Starting at the topmost node, the first question is the value of the feature “Outlook”. The answer is “sunny” and thus the sample moves along the branch marked “sunny” to the node on the right. Here the question is the value of “windy”. The answer is of course “false” and the sample end up in the leaf “no golf” and is classified as such.

Growing a decision tree means making it learn the right questions such that it can split the samples most efficiently into the available categories. This is done by following the classification and regression tree (CART) algorithm [4]. The algorithm aims to find the optimal split, meaning the value of one feature, that best separates the classes for the samples in that node. The procedure is illustrated in algorithm (1).

The Gini impurity (2.1) was chosen as a quantification of the split quality. K is the number of features

Algorithm 1: CART

```
1  sets = one set containing all samples;
2  while split quality improves do
3      for all sets do
4          Select rule that best splits the classes in the set;
5          for all channels in the set do
6              calculate value of chosen criterion;
7          end
8          split at channels with lowest/highest criterion value;
9          sets = new sets
10     end
11 end
```

and f_i the fraction of samples of class i in the considered set.

$$I_G(f) = \sum_{i=1}^{K} f_i(1 - f_i) \tag{2.1}$$

The algorithm thus aims to minimize the Gini impurity at each node.

Figure (2.2) shows a example of a decision tree in a random forest as grown is this project.

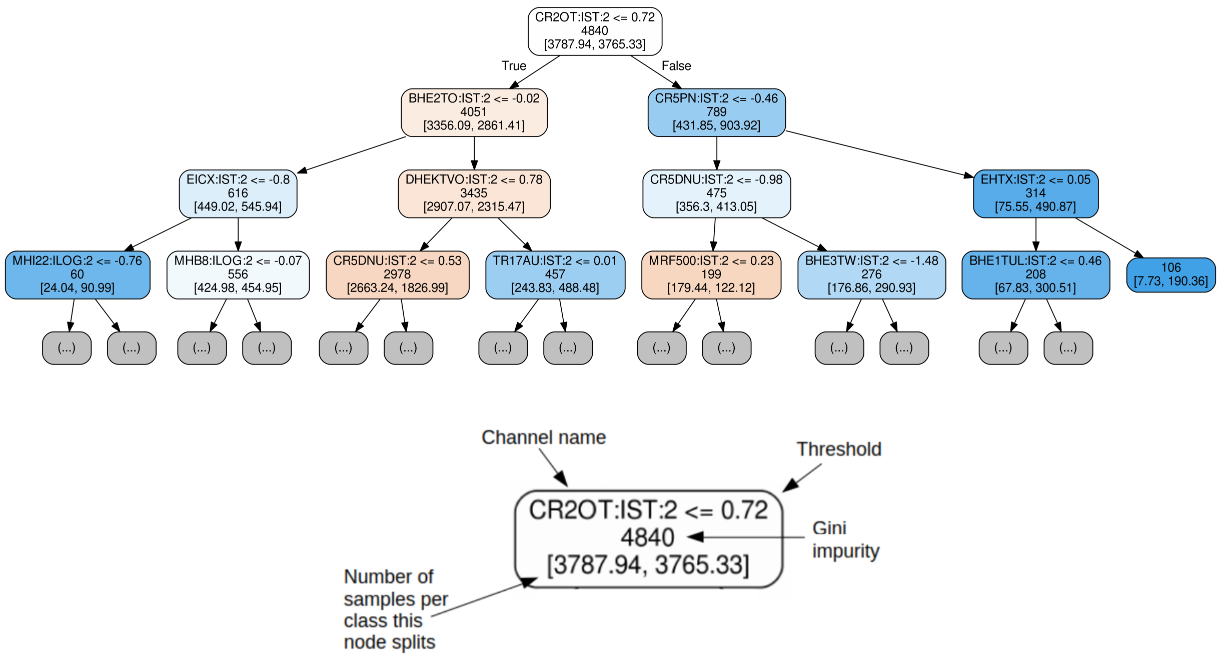


Figure 2.2: Above: Example of a decision tree in a Random Forest. Below: Explanation of the parameters in a decision tree node. The color of a node visualizes its purity. Here a deeper blue represents a high percentage of interlock samples, deeper orange a higher percentage of stable samples.

In a random forest the label of a new sample is generated by taking the average of the classification scores given by each tree for that sample. A random forest has a range of advantages over a single decision tree. As single trees in the forest do not see all the samples and features, they are de-correlated and thus overfitting is rarely an issue. Normalization of the input data is also not necessary as the Gini coefficient is not affected by scaling.

A inbuilt feature selection method of decision trees and thus random forests is feature importance ranking. In the scikit-learn RandomForestClassifier used in this project the importance score of a feature is calculated as the total decrease in node impurity, for nodes using that feature as a decision rule, weighted by the probability of a sample reaching the node. This probability is approximated by the proportion of samples in the node.

2.2 Convolutional neural network

The term convolutional neural network (CNN) designates a neural network (NN) that employs convolutions instead of general matrix multiplications. A convolution is a linear operation of the form

$$s(t) = (x * w)(t) = \int x(a)w(t-a)da \tag{2.2}$$

Here w is called a kernel and the output $s(t)$ is called a feature map.

In the case of two dimensional input data, as for instance, for image classification, the kernel would also be two dimensional and the convolution takes the following form

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.3}$$

with I the input image and K the applied kernel. Note that (m,n) do only run over a neighborhood of (i,j), not over the whole picture [5]. Figure (2.3) shows an example of such an operation.

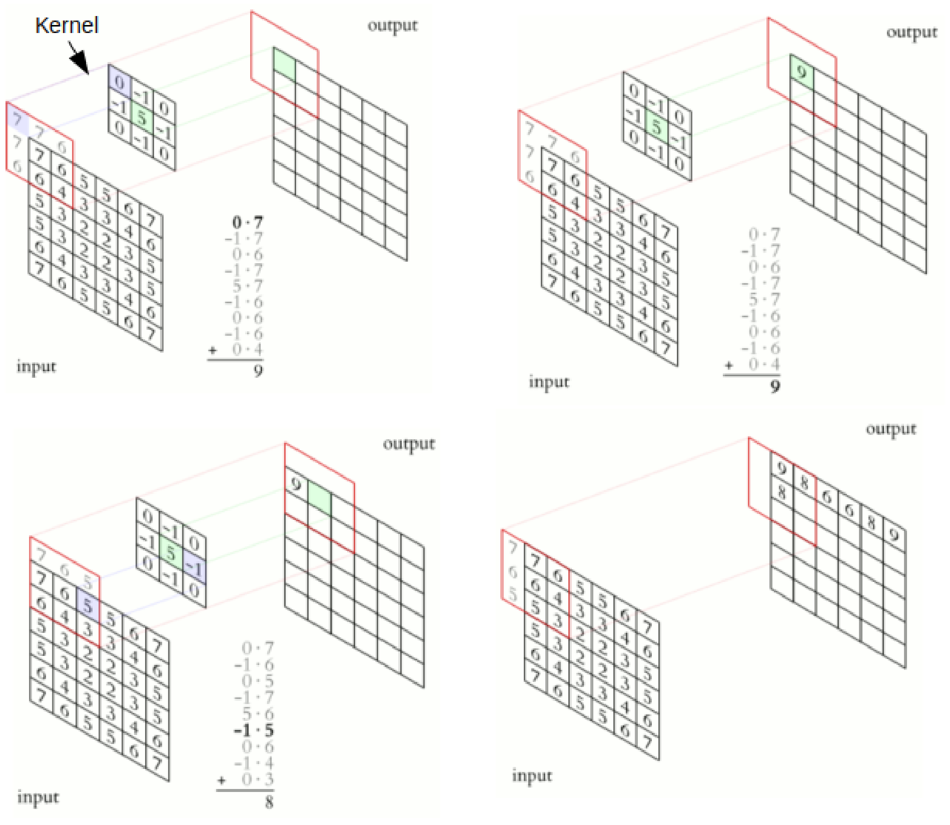


Figure 2.3: Illustration of a convolution. The kernel moves over the image and convolutes each input area into one output value [6].

The three main concepts leveraged in convolutional layers are sparse interactions, parameter sharing and equivariant representations.

Convolutional layers have sparse connectivity meaning that each input unit is only connected to a subset of the output units. As a consequence the number of parameters of the network and thus the number of operations needed to compute the output can be significantly reduced.

The weights of the kernel are used at multiple positions of the input. This reduces again the number of parameters the network has to learn.

Parameter sharing also makes the convolutional layer equivariant to translations of the input. For image data this means that the layer can detect a specific pattern at any location in the image.

Pooling and dropout are techniques used to normalize the output of a previous layer. Pooling consists of replacing the output of the layer by some summary statistic of nearby outputs. The max pooling employed in this project for instance, replaces the values of a determined region of the previous layers output by the maximum value of that region.

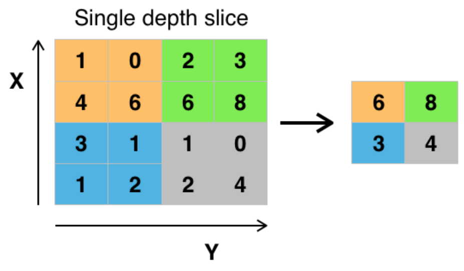


Figure 2.4: Illustration of a max pooling operation [7].

Dropout is another common regularization method. Here a randomly chosen proportion of the input units to the layer are set to zero. Note that dropout is only active during the training phase.

2.3 Shapley additive explanations

Interpreting the output of a complex machine learning model is a crucial part of the development but is often not straight forward. A number of model interpretation methods,such as LIME, DeepLIFT, Layer-Wise Relevance propagation, shapley regression values, shapley sampling values and quantitative input influence, have been developed over the years but it is not clear when to prefer one over the other.

SHapley Additive exPlanations (SHAP) by Lundberg and Lee [8] is the unification of six model interpretation methods, namely LIME, DeepLIFT, Layer-Wise Relevance propagation, shapley regression values, shapley sampling values and quantitative input influence. SHAP builds an "explanation model", i.e. a more easily interpretable approximation of the original model and quantifies the effect each feature has on the classification of a particular sample.

Figure (2.5) illustrates the explanation model concept.

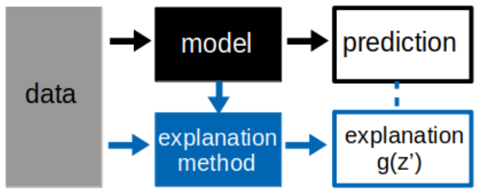


Figure 2.5: Illustration of the explanation model concept. Figure adapted from [9]

The authors define a class of "additive feature attribution methods"and show that the six considered methods all belong to this class.

Additive feature attribution methods are of the form

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i \tag{2.4}$$

with $z \in \{0,1\}^M$ and M the number of simplified input features.

The explanation model $g(z')$ attributes an effect ϕ_i to each simplified input feature z'_i. The model then approximates the output of the original model by summing over all simplified input features. Methods belonging to this class have a unique solution with three desirable properties. Local accuracy, missingness and consistency. Local accuracy means that for an original input instance x, with a corresponding simplified input z': $g(z') = f(x)$. Missingness means that if a features is missing from the original input it will have an attributed effect of zero in the explanation model. The consistency property states that changes in the original model that lead to the increase, or stagnation, of the contribution of a simplified feature z_i cannot lead to the decrease of the attributed effect ϕ_i.

A game theoretical proof for the existence of a unique solution for all member methods of this class is given, based on cooperative game theory and the three properties outlined above and introduce SHAP values as generalized feature importance measures.

The authors also produced a video, explaining the concepts in an easily understandable manner, which can be found at [9] at the time of writing.

The SHAP values obtained from the explanation model quantify the effect of a feature to the classification outcome of a sample as illustrated in figure (2.6).

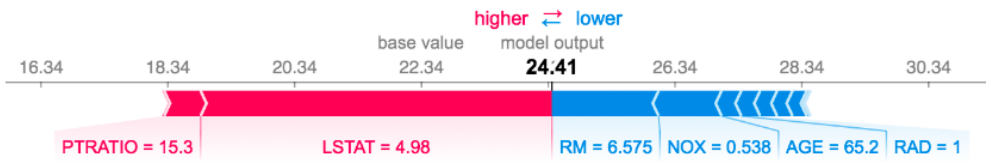


Figure 2.6: Force plot illustrating the interpretation of SHAP values ([10]).

A feature marked red “pushes”the prediction for the sample towards higher values, a feature marked blue “pushes”the prediction towards lower values. The SHAP values for each feature indicate the magnitude of the pushing. In this example the feature LSTAT has the highest impact on increasing the predicted value for this sample.

The effect of the features over all samples in a dataset can be visualized by a summary plot as seen in figure (2.7).

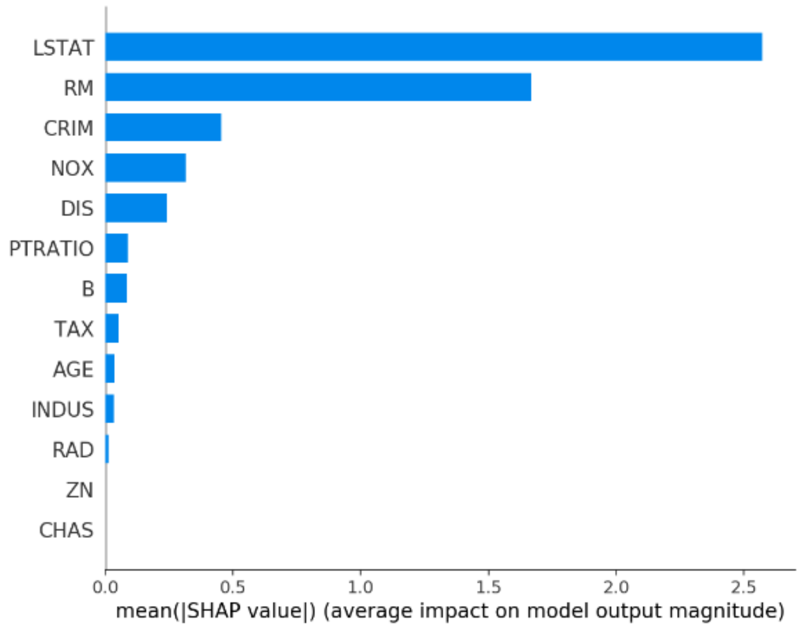


Figure 2.7: Summary plot illustrating the interpretation of SHAP values over a dataset ([10]) in case of a regression model.

In this example, the feature LSTAT does indeed have the highest impact on the predictions.

The SHAP package includes a range of explainer models specifically designed to different types of models. In this project we employed the SHAP Tree explainer [11], which is a high-speed variation optimized for tree ensemble methods. Other variations are the DeepExplainer and the GradientExplainer designed for NN, a LinearExplainer for linear models and the KernelExplainer which is described as a general, model agnostic method.

2.4 Recurrence plots

Recurrence plots were developed as a method to analyze dynamical systems and detect hidden dynamical patterns and nonlinearities [12].

In this project recurrence plots are used to transform our time series into images, that we can then classify using a convolutional neural network.

The original recurrence plot described in [12] is defined as

$$R_{i,j} = \theta(\epsilon_i - ||\vec{x}_i - \vec{x}_j||), \;\; \vec{x}_i \in \mathbb{R}^m, i, j = 1, \ldots, N.$$

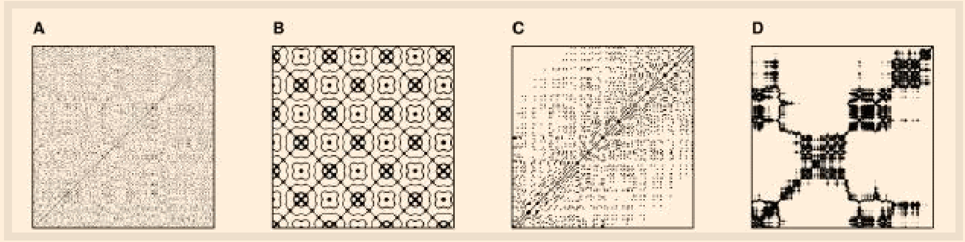
with θ the heavyside function. Here the radius ϵ_i is chosen for each i in such a way that the neighborhood it defines contains a fixed number of states $\vec{x}_j$. As a consequence the recurrence plot is not symmetric but all columns have the same recurrence density.

The most common definitions use a formulation with a fixed radius $\epsilon_i = \epsilon, \forall i$.

The variation we use here is a so called global recurrence plot [13] with a fixed epsilon.

$$D_{i,j} = \begin{cases} ||\vec{x}_i - \vec{x}_j||, & ||\vec{x}_i - \vec{x}_j|| \leq \epsilon \\ \epsilon, & ||\vec{x}_i - \vec{x}_j|| > \epsilon \end{cases}$$

D is symmetric. The structures in a recurrence plot contain information about the time evolution of the system it portraits. On a large scale its evolution can be characterized as homogeneous, periodic, drift or disrupted as shown in figure (2.8).



Characteristic typology of recurrence plots: (A) homogeneous (uniformly distributed noise), (B) periodic (super-positioned harmonic oscillations), (C) drift (logistic map corrupted with a linearly increasing term) and (D) disrupted (Brownian motion)· These examples illustrate how different RPs can be· The used data have the length 400 (A, B, D) and 150 (C), respectively; no embeddings are used; the thresholds are $\varepsilon = 0.2$ (A, C, D) and $\varepsilon = 0.4$ (B)·

Figure 2.8: Illustration depicting examples of characteristic topologies of recurrence plots taken from[14]

Recurrence Quantification Analysis (RQA) deals with the analysis of the small scale structures in recurrence plots. RQA presents a number of metrics to quantify the recurrences, such as the recurrence rate, trapping time or divergence. RQA can without issue be applied to non-stationary or very short time-series as are present in this project.

The patterns of recurrence plots thus contain a wealth of information not directly available from the time series they are based on. This information may be extracted by a NN as we will demonstrate shortly.

Chapter 3

Correlation Analysis

The following section is concerned with the investigation of the correlation between channels. The information obtained from the analysis of remarkably high and low correlations in particular may be useful for the selection of a minimal feature set for the considered models.

Figure (3.1) displays the correlations between the channels that were selected by expert opinion as discussed in the introduction.

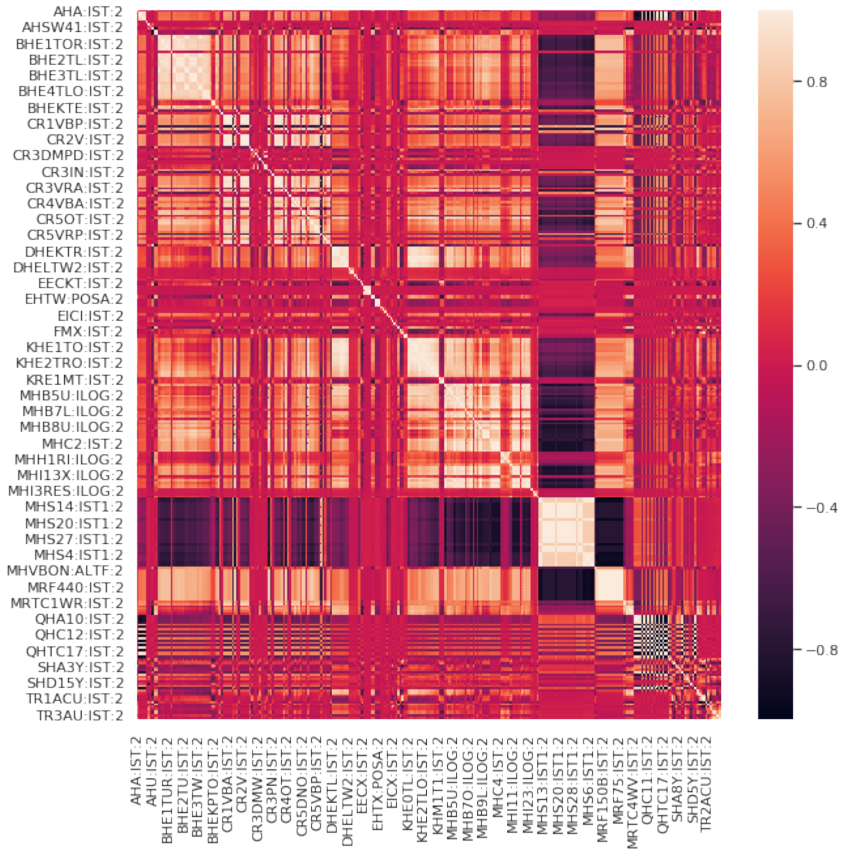


Figure 3.1: Heatmap of the correlations between channel for the 2019 Data.

Note that not all channel names are indicated on the axis. The displayed channels are ordered alphabetically and are meant to give an indication to which channels the displayed correlation values belong. As the channels are ordered alphabetically channels that record similar signals are grouped together. Thus one observes areas of highly correlated channels along the diagonal.

In the “MHS..”¡ channel region near the bottom of figure (3.1) one observes a band of highly negative correlation values. These channels may be interesting as relevant number of them might be removed without loss of information. Figure (3.2) displays a zoom in of figure (3.1) to this region.

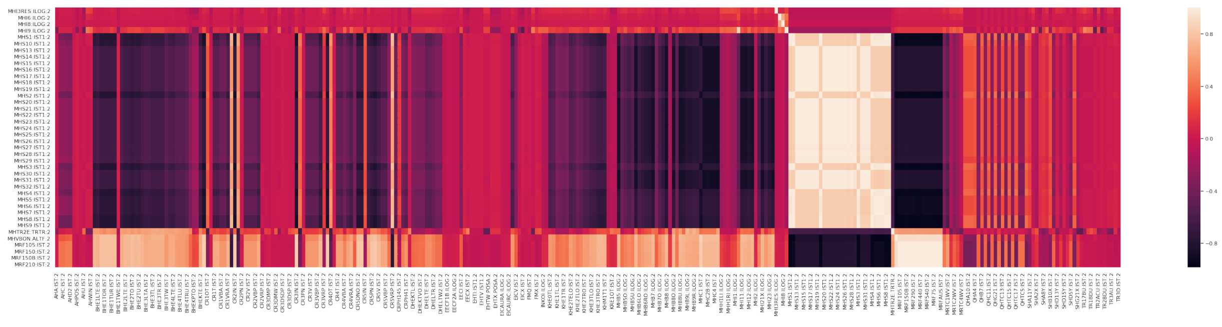


Figure 3.2: Zoom in of the correlation heatmap displayed in Fig.(3.1).

The channels producing the band are the channels from MHS1:IST1 to MHS9:IST1:2 which are all center of gravity monitors (Schwerpunksmonitore) X/Y (horizontal/vertical). One may want to pay attention to these channels in the feature selections as they may potentially be removed from the set without loss of information.

Figure (3.3) displays the values from figure (3.1) but in absolute values and clipped at correlation values of 0.9. This means that only areas displaying very high positive or negative correlations are colored.

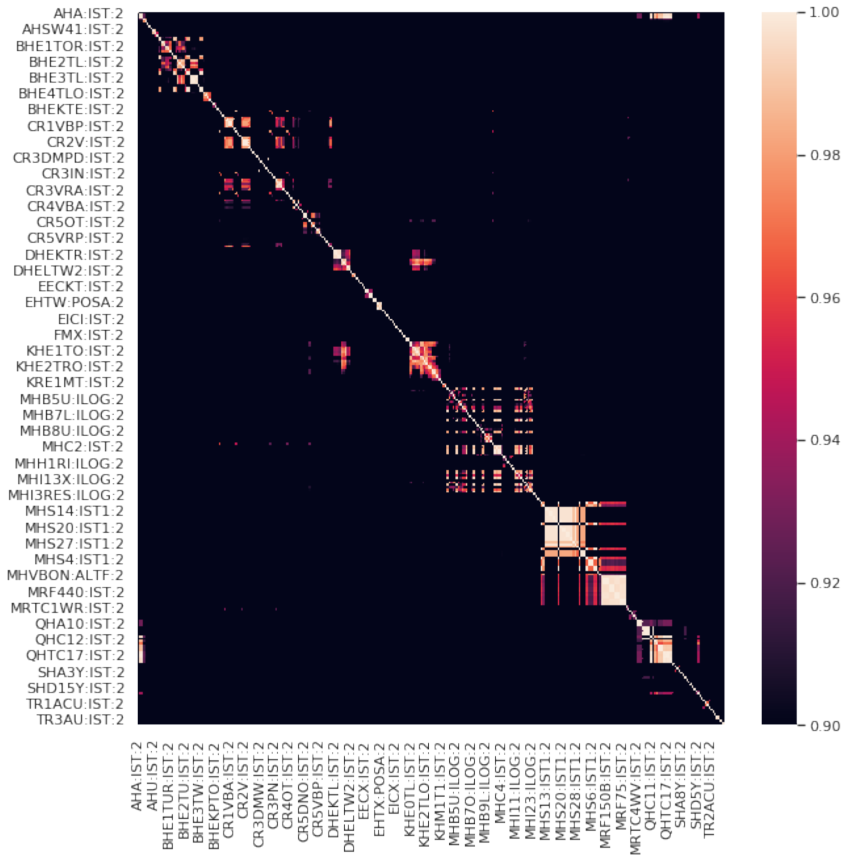


Figure 3.3: Heatmap of the correlations between channel for the 2019 Data for high correlation values. The values have been thresholded at 0.9 and only the absolute values are displayed.

As expected, one mostly observed regions along the diagonal. The plot also shows two interesting off diagonal regions displayed as zoom-ins in figure (3.4).

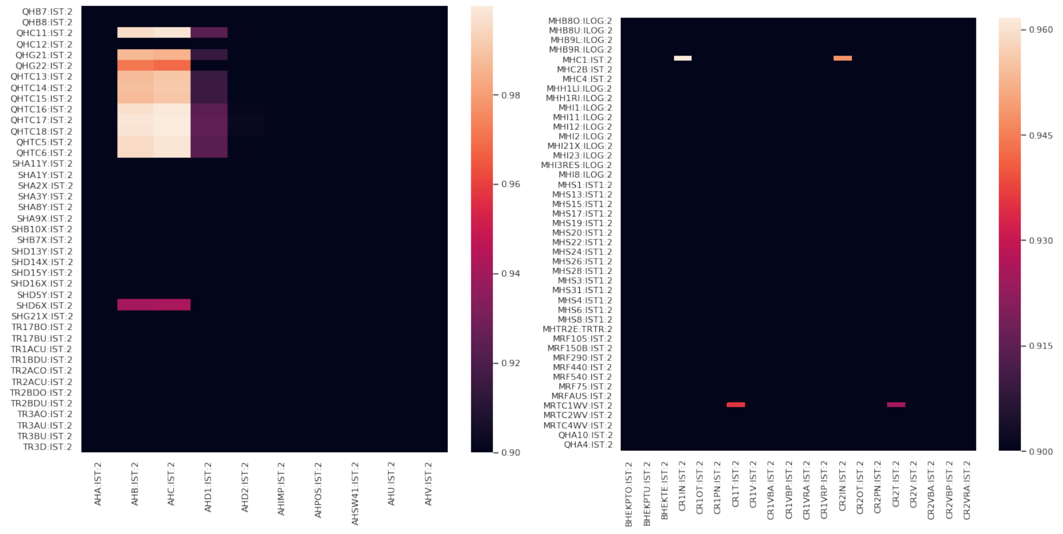


Figure 3.4: Zoom in of two regions of the correlation heatmap displayed in figure(3.3). All values have been thresholded at 0.9.

One sees a few potentially interesting correlations between individual channels. The SHD6X:IST:2 channel seems to be highly correlated with the channels AHB:IST:2 and AHC:IST:2. Something interesting also seems to happen in the relations between channels QHC11:IST:2 to QHTC6:IST:2 and channels AHB:IST:2 to AHC:IST:2. From the left side plot it appears that the channels MHC1:IST:2 and MRTC1WV:IST:2 are highly correlated to the channels CR1N:IST:2, CR2IN:IST:2 and CR1T:IST:2, CR2T:IST:2 respectively. One may want to pay closer attention to these channels during the feature selection process.

Figure (3.5) displays the values from figure (3.1) but clipped at correlation values of 0.01, thus showing only regions that barely correlate.

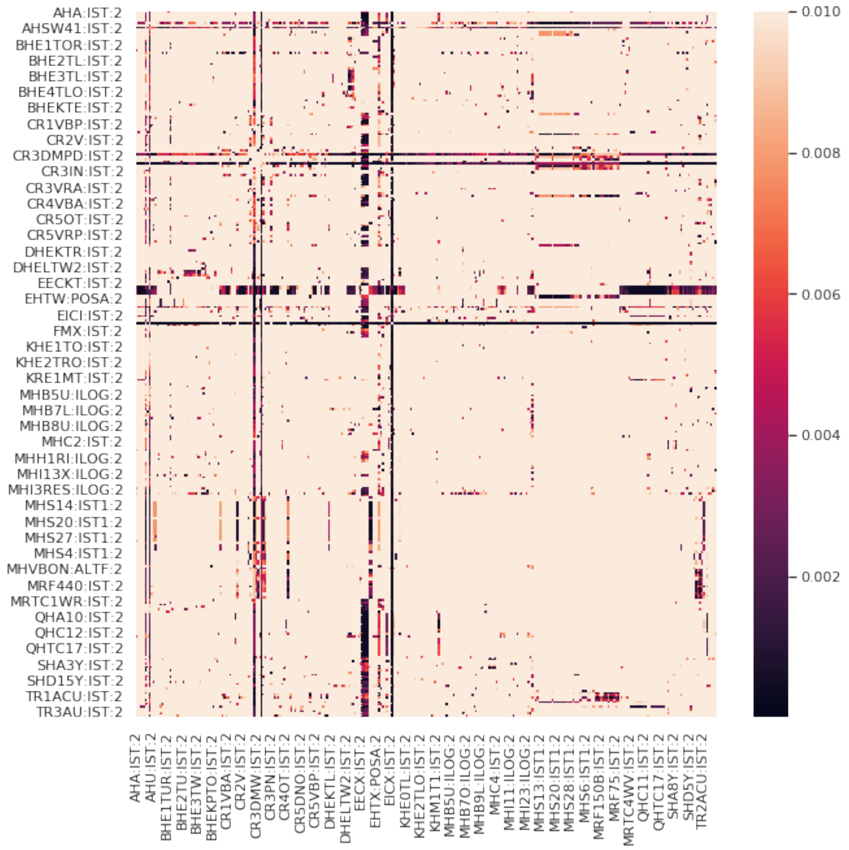


Figure 3.5: Heatmap of the correlations between channels for the 2019 Data for low correlation values.The values have been thresholded at 0.01.

It shows a few interesting regions as well as many individual channel interactions that may be of interest. At this point the focus will remain on channels that display low correlations with all other features. Figure (3.6) displays regions from the low correlations plot in figure (3.3).

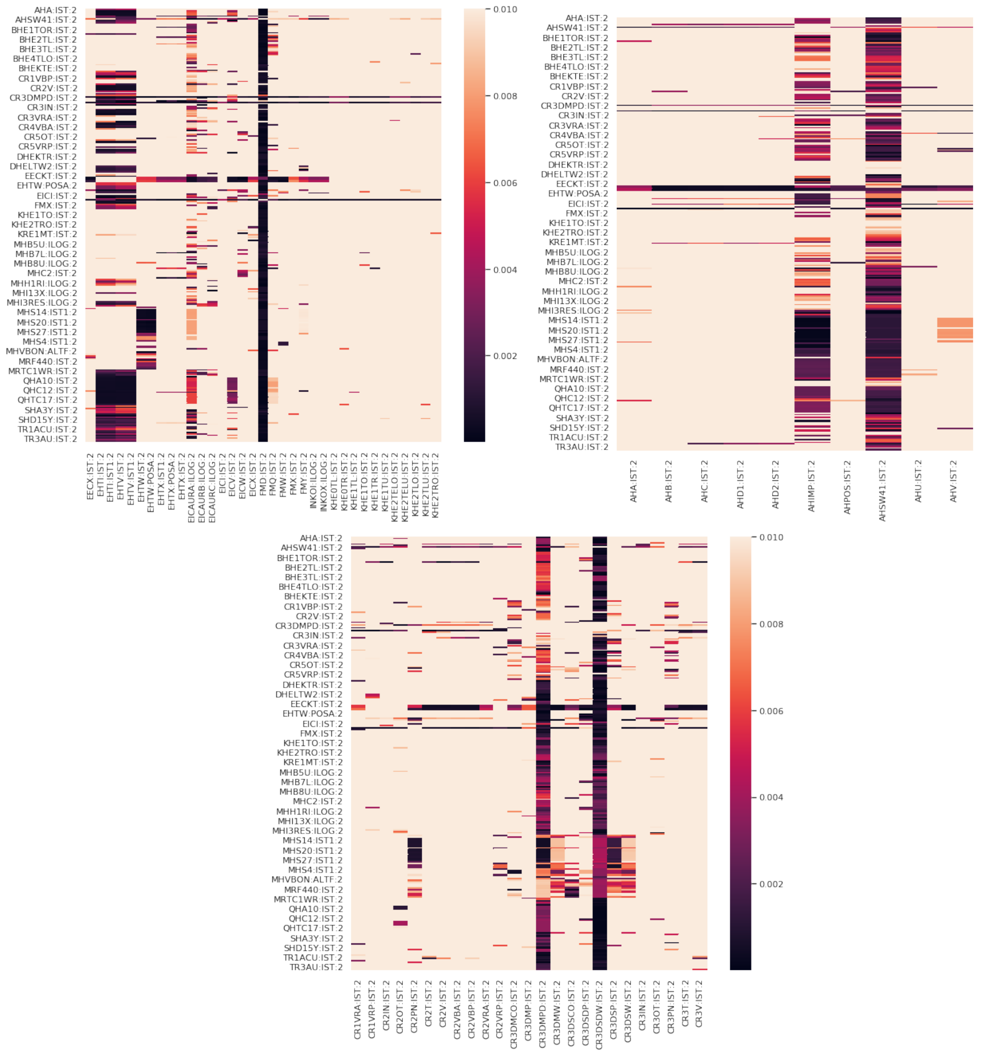


Figure 3.6: Zoom in of two regions of the correlation heatmap displayed in figure(3.3). All values have been thresholded at 0.01.

Channels displaying very low correlations with all other channels could contain information not present in other channels. One may want to pay attention to these channels in the feature selection. If they do contain information relevant to the classification task one would expect them to be present in the selected feature sets. From the heatmaps displayed in figure (3.6) one can identify the channels FMD:IST:2 (focusing magnet dipole current), AHIMP:IST:2 (Combination for beam momentum through super button), AHSW41:IST:2 (Deflecting magnet for PIE5) as well as CR3DMPD:IST:2 (HIPA CR3 ring tuning) and CR3DSDW:IST:2 (also HIPA CR3 ring tuning).

In this chapter channels were identified that may be relevant during the feature selection process for the considered models. The channels identified as having high correlations with other channels in the set may potentially be removed safely from the a feature set without loss of information. The channels that were identified as having low correlations with most other channels in the set may contain relevant information and one should consider to add those to the feature set.

Chapter 4

Models

In the following section a random forest model and a CNN with recurrence plots model are investigated.

All models were implemented using the Python programming language. The MLlib framework library [15] developed for this project in collaboration with Renato Bellotti and Sichen Li was employed for ease of storage and reproducibility. MLlib was designed to provide a unified interface for the different machine learning projects at PSI in order to improve communication and knowledge transfer. All the implementation details of different libraries and model types are hidden in a corresponding surrogate class. The syntax for the end user is thus consistent and models using different libraries can be easily compared. The saved model folder also contains all the preprocessors used on the data. This makes the workflow transparent and easily reproducible. At the time of writing, MLlib supports the tensorflow [16] and scikit-learn [17] libraries.

Due to the nature of the channels, one suspects that a relevant number of them might be strongly correlated and that it should thus be possible to reduce the amount of channels needed for the classification task. Different feature selection methods have been employed to identify the most relevant channels for the interlock classification problem. From the original expert selection of 450 channels in the 2018 Dataset, 311 do not contain any NaN values in the 2019 dataset used for the results presented here. These 311 channels are the base features for the models presented below.

The ultimate aim of the project is to obtain a model that can be employed for online predictions in order to support the work of the beam line operators and reduce beam time loss. A GUI was implemented by Jaime Maria Coello de Portugal Martinez Vazquez to perform such live predictions. Online predictions mean rolling window predictions, as a new prediction has to be made for every new time step. One can simulate this procedure by doing rolling windows predictions on the test set.

ROC curves as well as the metrics target value, AUC and beam time lost, as defined in in equations (1.2), (1.1) and (1.3) respectively, will be used to evaluate the model performances. As the target value and the beam time lost do essentially contain the same information, beam time lost will be indicated for the main results only.

Notebooks for all experiments presented below can be found and accessed in the following repository: `https://gitlab.psi.ch/zacharias_m/masterthesis`

4.1 Random Forest

The Random Forest classifier is a widely used ensemble method appreciated for its performance as well as its easy interpretability. The performance of this model is evaluated on the time-series classification task.

4.1.1 Methodology

The dataset was split into a training, validation and test set where the training set are the first 70% of the dataset and the validation and test set the two remaining 15% in time order.

The random forest model presented here was build using the RandomForestClassifier from the scikit-learn library and the SklearnSurrogate class from the MLlib library.

The model configuration was chosen by grid search with AUC as the tracked metric. A random forest classifier with 90 estimators, a maximal depth of 9 and maximal number of leaf nodes of 70 with Gini impurity as the splitting criterion was found to be the best performing variation.

The time series is loaded from file then cut into windows as shown in figure (4.1).

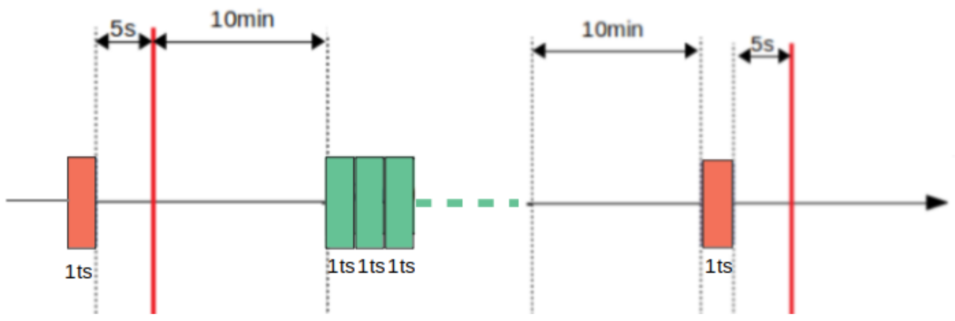


Figure 4.1: Illustration of the input data for the random forest model. The windows are cut to a length of one timestamp.

Windows of a lager size were also investigated. These windows have to be flattened before being handed to the classifier. The procedure is illustrated in figure (4.2).

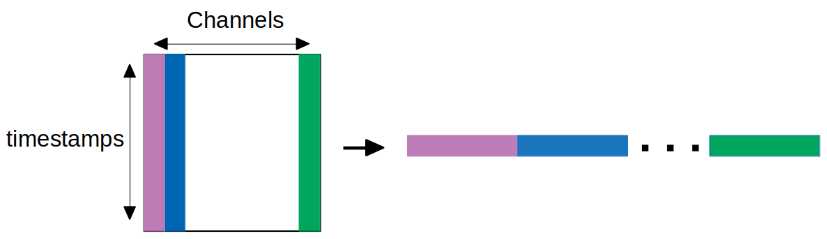


Figure 4.2: Illustration of window flattening.

4.1.2 Results

All the experiments have been performed on the 2019 data with randomized interlock timestamps. The confidence intervals have been calculated by 10-fold bootstrapping over the training and validation sets. Note that the confidence intervals in the ROC curves apply to the TPR. They are thus vertical errors, not threshold errors. The confidence intervals given for the metrics were obtained as two standard deviations of the corresponding metric.

Table (4.1) contains the target values, AUC and the beam time lost, as defined in equations (1.2), (1.1) and (1.3), for the shuffled and time ordered models. Figure (4.3) shows the ROC plot obtained for the Random Forest for shuffled and time ordered windows in the training set.

	time ordered	**shuffled**
Target value	0.26 ± 0.02	0.26 ± 0.03
AUC	0.72 ± 0.02	0.72 ± 0.02
Beam time lost [s/interlock]	20.5 ± 0.4	20.5 ± 0.5

Table 4.1: Maximal distance to the target for the ROC curves obtained with models trained on shuffled and timeordered input windows.

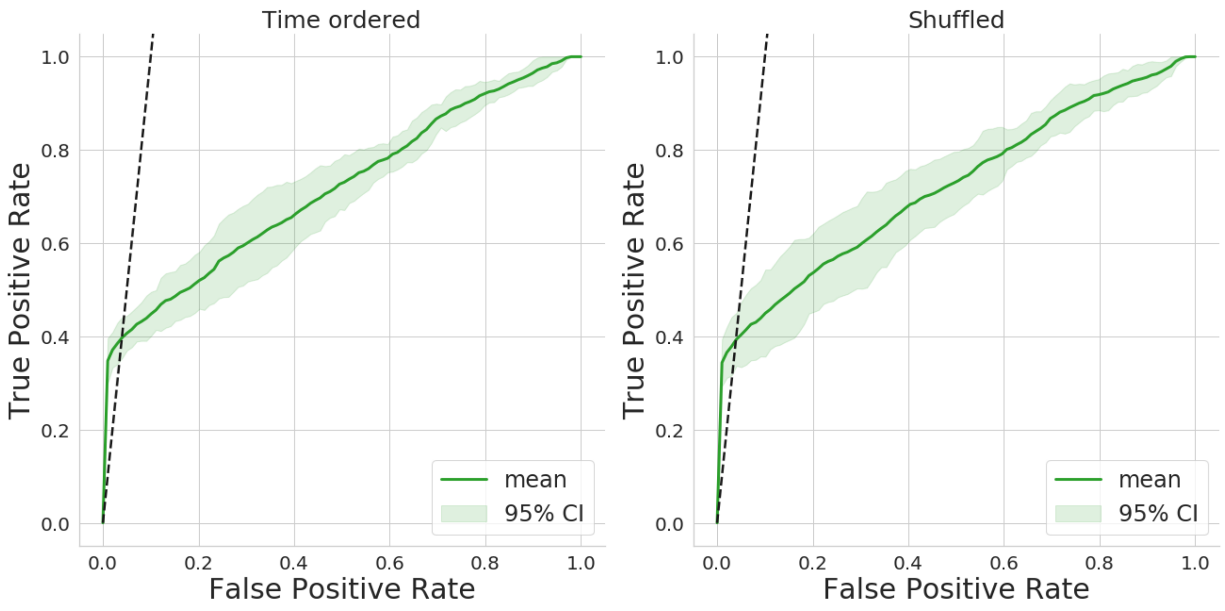


Figure 4.3: Comparison of the performance of the Random Forest model for time ordered and shuffled training samples. Both models were trained on windows of length one. The dotted line refers to the target to maximize equation(1.2)

In both cases the model was trained on all the stable and interlock windows in the training set and evaluated on all windows in the validation set. Note that shuffling was applied to the training set but not to the validation set.

No significant difference can be observed in the metrics for the two models. The slightly higher standard deviation of the target value and beam time lost obtained for the shuffled model may be attributed to the limited number of bootstrapped runs.

The performance of the model trained on stable windows from different positions in the dataset was analyzed and the results are presented in figure (4.4) and table (4.2).

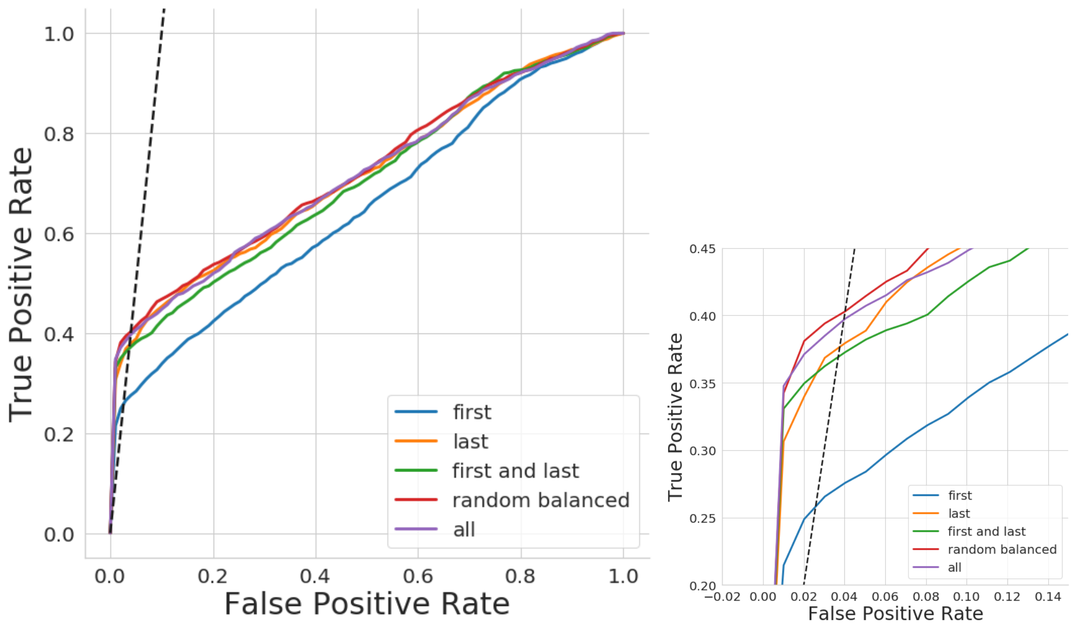


Figure 4.4: Left: Mean ROC curves over three bootstrapped runs for the model trained on windows of length 1 time step. Right: Zoom in of the region near the target line.

	First and Last	**First**	**Last**	**Random balanced**
Target value	0.25 ± 0.03	0.14 ± 0.04	0.23 ± 0.03	0.26 ± 0.03
AUC	0.70 ± 0.02	0.65 ± 0.02	0.71 ± 0.03	0.72 ± 0.02

Table 4.2: Maximal distance to the target for the ROC curves obtained with models trained on stable windows of different position relative to the interlock events. “First” denotes the first stable windows after an interlock event, “Last” the last stable window before an interlock event. “All” means that all available stable windows have been used. “Random balanced” indicates that randomly chosen samples were used whose number equals the number of interlock samples.

The ROC curves presented in figure (4.4) are the mean curves calculated over all bootstrapped runs. The variation trained on the “first” stable windows achieves the lowest scores, while the AUC as well as the target value of all other variations are in a comparable range.

Figure (4.5) compares the performances of the model for different window sizes. The target values for the variations are listed in table (4.3).

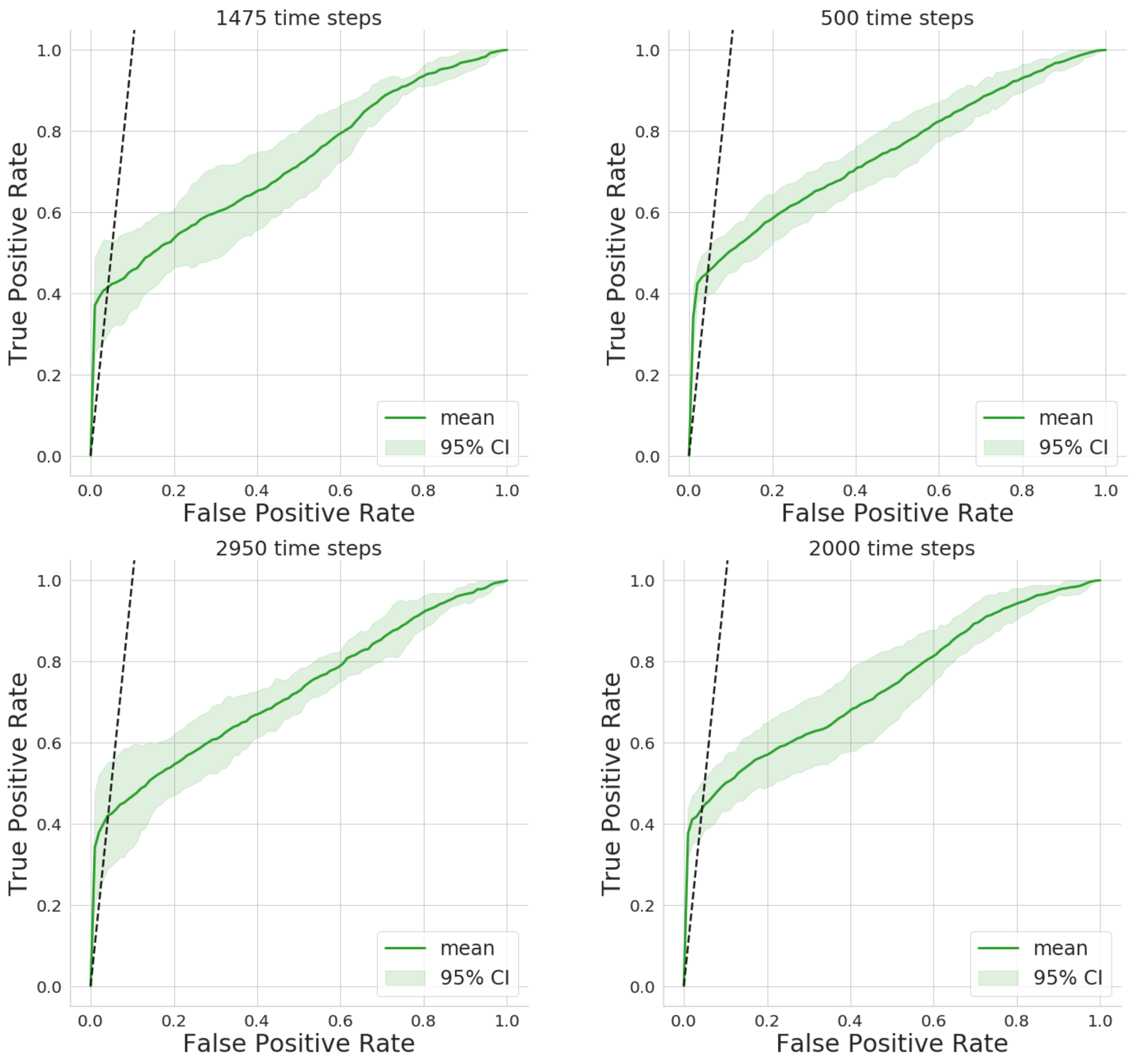


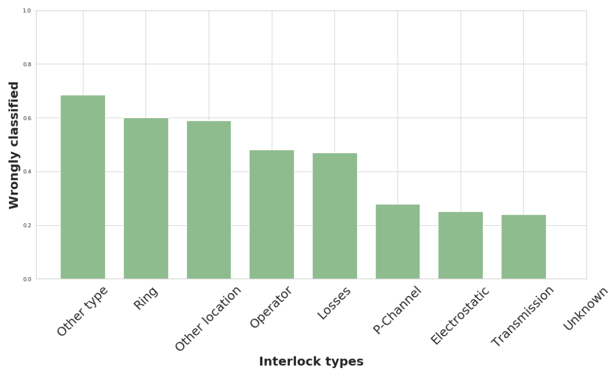
Figure 4.5: Performance of the Random Forest model over different sizes of the input window.

	2000 ts	**1475 ts**	**500 ts**	**2950 ts**
Target value	0.31 ± 0.03	0.31 ± 0.05	0.27 ± 0.03	0.28 ± 0.06
AUC	0.74 ± 0.03	0.72 ± 0.03	0.75 ± 0.02	0.72 ± 0.02
Beam time lost	19.4 ± 0.5	19.5 ± 0.9	20.4 ± 0.7	20.0 ± 1.0

Table 4.3: Maximal distance to the target for the ROC curves obtained with flattened windows of different length.

The performances for windows of length 500, 1475, 2000 and 2950 time stamps are compared. The model trained on windows of length 500 time stamps seems the most stable. The models trained on windows of length 500, 1475, 2000 and 2950 time steps all present comparable AUC and target values, though the 2000 and 1475 model may outperform the others in terms of target value. The number of runs is not sufficient to draw a conclusion.

Figure (4.6) presents the miss-classification rates by interlock types for the models trained on windows of length 1 time stamp and 2000 time stamps. Note that a “interlock-score”threshold of 0.5 was applied here.



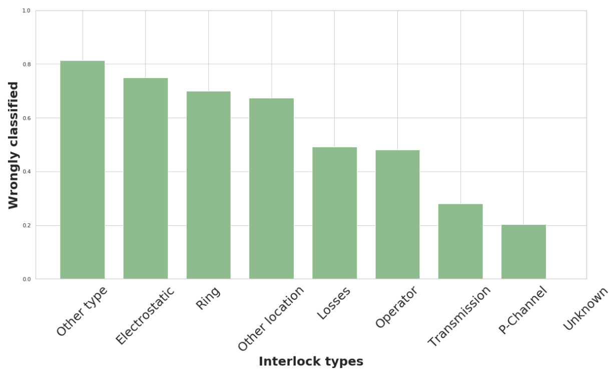


Figure 4.6: Performance of the Random Forest model for 1 time stamp (left) and 2000 time stamps (right) by interlock type

The relative miss-classification rates seem fairly similar for the two models, with the exception of type “Electrostatic” which is remarkably lower for the 1 time stamp model. The 2000 time stamp model appears to have slightly higher miss-classification rates for all types, except “Losses” were the miss-classification rate is about the same as for the 1 time stamp model and “P-channel” were it outperforms the 1 time stamp model.

Online predictions Figure (4.7) presents the performance of the random forest model trained on windows of length 1 on live interlock predictions. Note that these predictions have been made with the random forest model trained on the dataset containing non-randomized interlock time stamps.

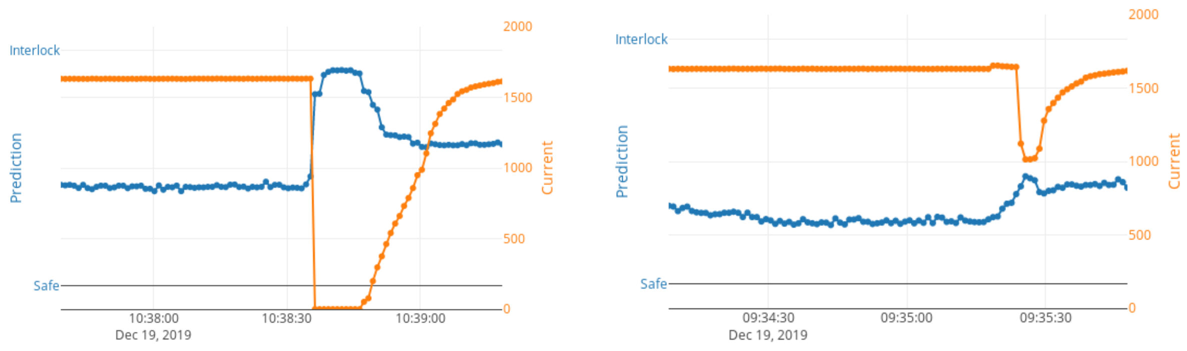


Figure 4.7: Performance of Random Forest model trained on windows of length 1. Left: Reaction of the model to an interlock event. Right: Reaction of the model in a stable region with a UCN kick. Figures by Jaime Maria Coello de Portugal Martinez Vazquez

The beam current is presented in yellow, the model predictions in blue. Time stamps with an interlock score below 0.5 (middle line) would be labeled “stable”, above “interlock”. The interlock scores do increase well above the 0.5 threshold during an interlock event. However the increase is only apparent on the time step of the event. In the stable region the interlock score stays below 0.5 for the entire region.

4.1.3 Feature Selection

Feature importance and mean SHAP values, as discussed in chapter 2.1 and 2.3, were explored for the random forest model with the configurations discussed above and windows of length 1 time stamp as input samples.

Feature Importance Figure (4.8) shows the 50 channels with the highest feature importance scores obtained with the settings detailed above.

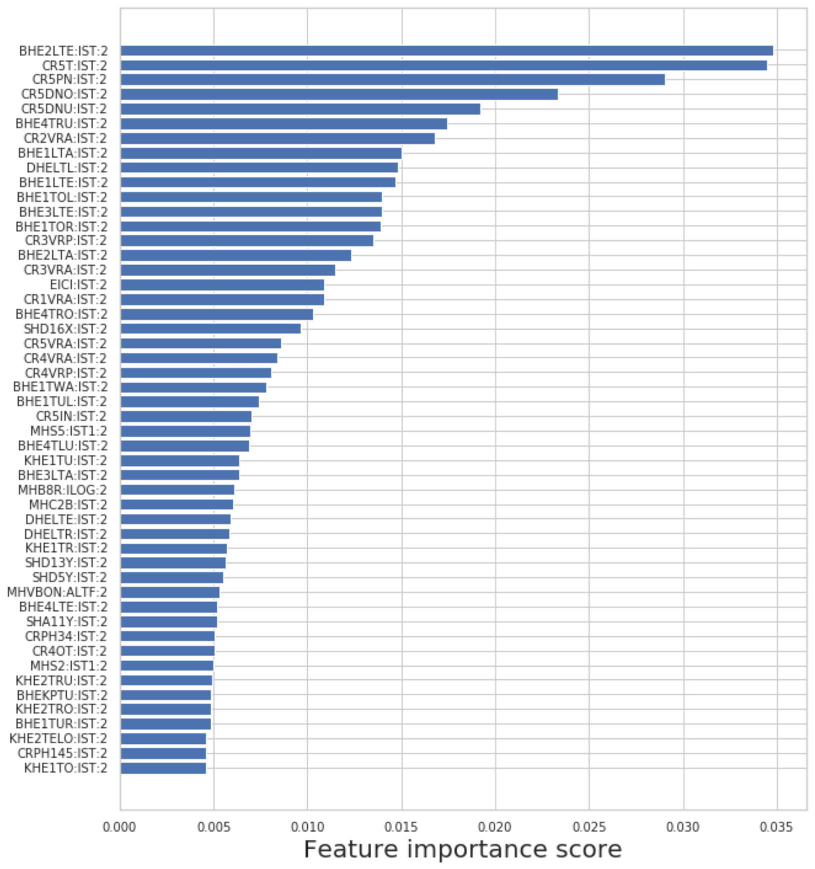


Figure 4.8: Feature importance values for one instance of a random forest model trained on windows of length 1 time stamp. Only the 50 highest values are presented here. See the Appendix for all.

The dominant contributions seem to come from four to five channels. Note that none of the feature importance scores reach a value higher than 0.035 and the majority of the scores rank below 0.02.

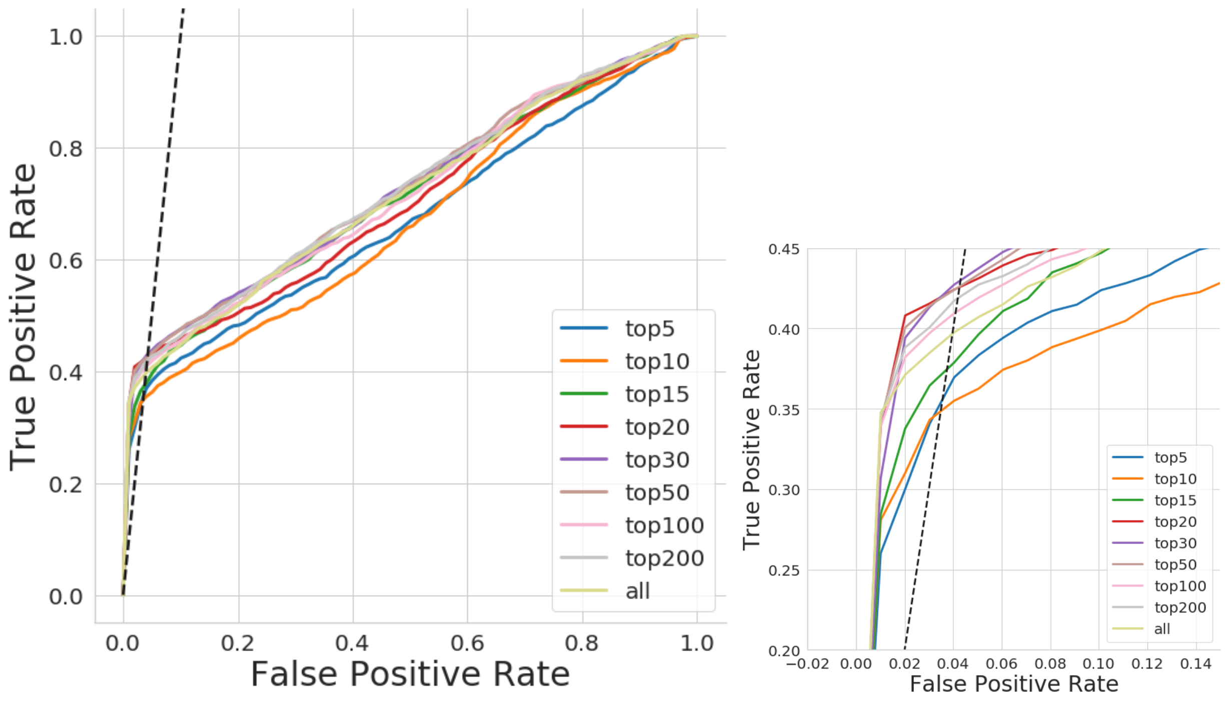


Figure 4.9: Models trained on a range of different features from the feature importance selection presented in figure (4.8).

	Top 5	Top 10	Top 15	Top 20	Top 30	Top 50	Top 100	Top 200
Target value	0.19 ± 0.02	0.21 ± 0.04	0.20 ± 0.03	0.26 ± 0.03	0.24 ± 0.03	0.26 ± 0.04	0.26 ± 0.04	0.26 ± 0.02
AUC	0.67 ± 0.01	0.67 ± 0.03	0.71 ± 0.02	0.70 ± 0.02	0.72 ± 0.02	0.72 ± 0.02	0.71 ± 0.03	0.72 ±0.03

Table 4.4: Maximal distance to the target for the ROC curves obtained using the respective features.

The variation trained on only the top 5 channels obtained from the feature importance ranking presents AUC and target value metrics only slightly below the variation trained on all 311 channels. All other variations show results comparable to the "all" variation. Note that the model trained on the top 10 features seems to show a lower AUC value as the top 5 model and the top 20 model seems to achieve a larger target value than the top 30 model.

Shapley additive explanations The model was trained on the data from the previously defined training and validation set. The SHAP tree explainer was used to build an explainer model and the mean SHAP values were calculated over all the samples in the training set. Figure (4.10) shows the channels with the 50 highest mean SHAP values.

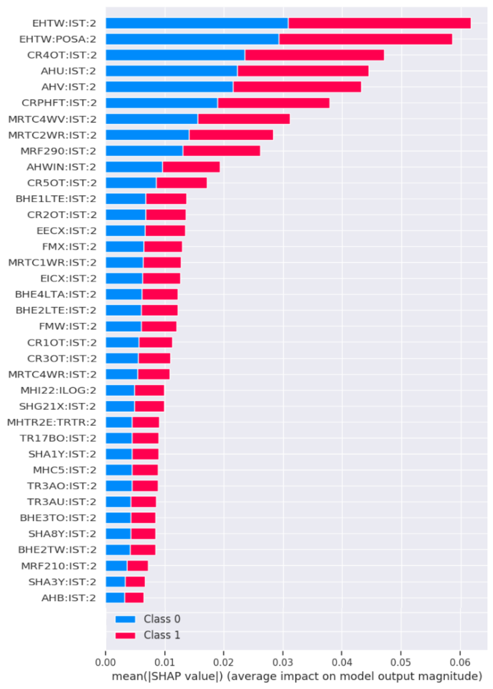


Figure 4.10: Mean SHAP values for the Random Forest model for the last and first stable windows in the training set and all windows in the validation set.

From figure (4.10) eleven channels seem to be most impact full for sample classification, though it is worth to note that the highest mean SHAP value is merely 0.06. The bottom 400 channels have a mean SHAP value of, or close to, zero and thus seem to have no impact on the classification. The full ranking can be found in the appendix.

The ROC curves presented in figure (4.10) are the mean over four runs obtained in the manner described above. The maximal distances to the target for each configuration can be found in table (4.5).

	Bottom 100	**Bottom 300**	**top 80**	**Top 50**	**Top 20**	**Top 5**
Target value	0.17 ± 0.03	0.28 ± 0.03	0.28 ± 0.03	0.19 ± 0.03	0.14 ± 0.03	0.08 ± 0.2
AUC	0.66 ± 0.2	0.73 ± 0.03	0.71 ± 0.02	0.70 ± 0.02	0.65 ± 0.03	0.57 ± 0.02

Table 4.5: Maximal distance to the target for the ROC curves obtained using the respective features.

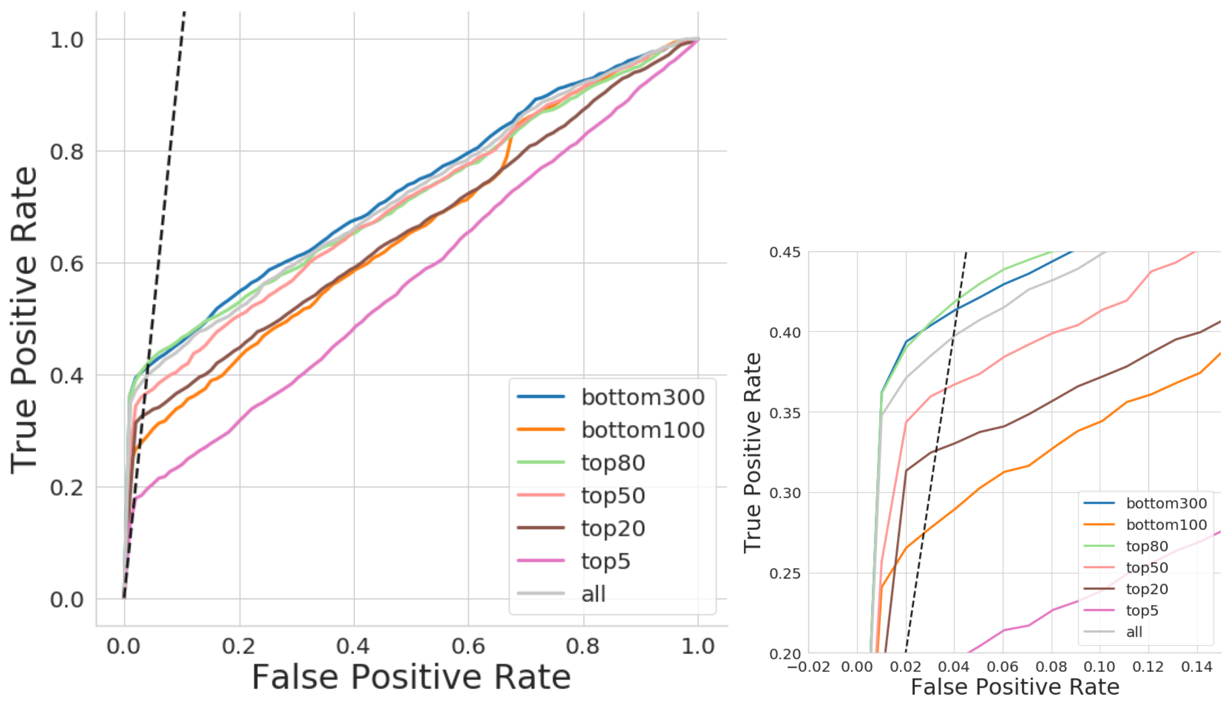


Figure 4.11: Random Forest models trained on windows of length 1 time stamp and various sets of the SHAP features.

In terms of the maximal distance to the target, the model trained on only the topmost 5 features achieves a performance comparable to the score obtained with a model trained on the bottom 100 features and higher than the score achieved with the bottom 200 features. The score obtained with the top 10 features of is comparable to the score obtained with all features though this score is only reached by the bottom 300 features model. However, when we also consider the ROC curve, it becomes apparent that only the bottom 300 model reaches the performance of the model trained on all features. The top 5, as well as the top 10 model present a ROC curve lower over all FPRs than the rest of the considered models. Note that all the above statements have to be taken with caution as long as only rudimentary confidence intervals have been obtained.

4.1.4 Discussion

No relevant difference in performance was found between a model trained on shuffled and time ordered data. This result was expected as the interlock distributions are fairly consistent over the whole dataset. The random forest model, in the formulation presented here, does not exploit time dependent trends in the data thus we would expect the same performance if the training data is shuffled or not.

For the random forest model the position of the stable windows seems to be irrelevant as the model appears to extract the same information from all instances. Models trained on first and last stable windows show comparable performance to the model trained on all stable windows or a random selection of stable windows. Flattened windows of length 500 time stamps present the least variability, although models trained on windows of length 2000 and 1475 time stamps may show slightly higher AUC and target values. Both the 1 time stamp model and the 2000 time stamp model display varying classification performances for the different interlock types. This may indicate that the model would benefit from a multi-class formulation with one class label for every interlock type. The results obtained from the online predictions indicate that the model seems confident in prediction of stable windows but does react too late to interlock events. The model did react correctly to the UCN kick and classified it correctly as a “non-interlock” event. The online predictions demonstrate that, even though the model does recognize an interlock event, it does not predict it and is thus may be of limited use for practical purposes. This result may also suggest issues with the current evaluation methods as the conclusions drawn from the model evaluation differs from the online prediction result.

The obtained mean SHAP values and the corresponding model performances back the hypothesis that a significant amount of the channels are redundant. The model trained on only the top five channels already shows comparable performance to the model trained on all channels and the model trained on the top 50 channels achieves the metrics reached by the model trained on all channels. Models trained on the top 80 channels and on the bottom 300 channels slightly outperform the model trained on all features. This last observation suggest a measure of redundancy in the information contained in the channels.

The results obtained on the feature importance selections also indicate that the combination of channels does matter, as adding more channels to the set does not necessarily improve and may even worsen, the performance results.

The difference in behavior of the models trained on sets of the feature importance selections and the SHAP selections is interesting and may warrant further investigation.

4.2 Convolutional Neural Network with Recurrence Plots

The main concept behind the following CNN model is to translate our time series classification problem into an image classification problem. The method used here was inspired by work done by Nima Hatami et al. [18] and consists in transforming the time series into recurrence plots.

Two variations of recurrence plots were explored. In version one a separate recurrence plot was drawn for each time series, i.e. each channel, see figure (4.12). In version two one recurrence plot was created from all the time series in the window as shown in figure (4.13).

4.2.1 Methodology

The NN was built with the TensorFlow 2.0 Keras api. The dataframe is loaded using the OurArchiver-DataSource MLlib class and all channels containing any number of undefined values are discarded. At the moment of writing we remain with 311 channels which are listed in table (7.3) in the appendix. The data is then cut into windows of a specified size following the procedure outlined in the introduction and split into training validation and test sets. The samples are then reformatted into 3D arrays of the following shape: (number of samples, window length, number of channels). For the one plot variation the time series are normalized to mean 0 and standard deviation 1. For the one plot per feature variation no further preprocessing was applied as normalization of the data is not necessary given the nature of the recurrence plots.

The recurrence plots are produced in a custom tensorflow layer. This allows to produce the plots on the fly saving memory and simplifying the application of the model in the Live GUI. It also allows to make the ϵ parameter which defines the extend of the recurrence area a trainable parameter of the network.

In the second custom layer, called ZeroTriangular, the lower triangle of the recurrence plots is set to zero to reduce the amount of redundant information. In the formulation applied here the recurrence plots are symmetric about the main diagonal, thus one can do this without loss of information.

Figure (4.12) illustrates the preprocessing for the one plot per feature approach.

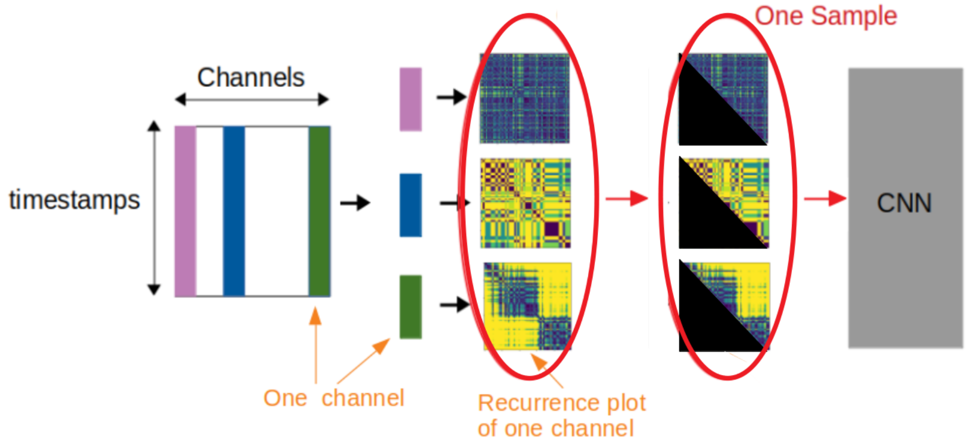


Figure 4.12: Illustration of the recurrence plot layer in- and output for the one plot per feature case.

Figure(4.13) illustrates the preprocessing for the one plot approach. The same procedure is applied here, though only on the one produced plot.

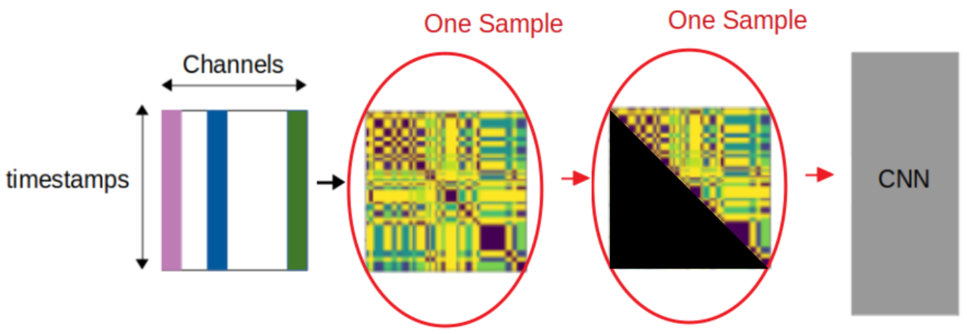
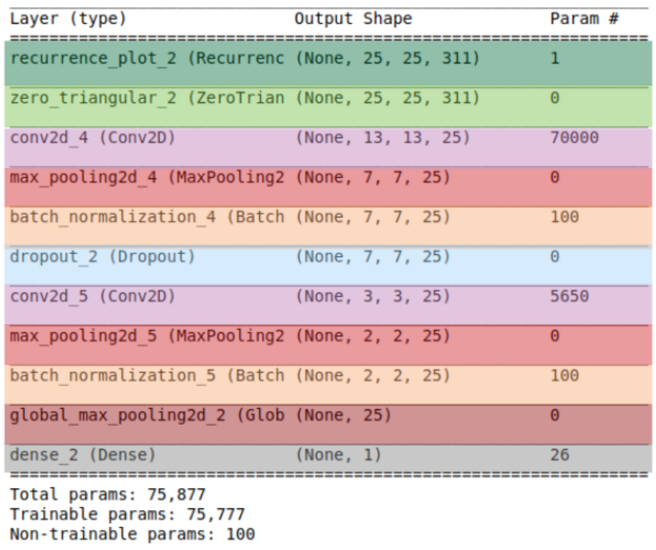


Figure 4.13: Illustration of the recurrence plot layer in- and output for the one plot case.



```
Layer (type)                 Output Shape              Param #
=================================================================
recurrence_plot_2 (Recurrenc (None, 25, 25, 311)       1
zero_triangular_2 (ZeroTrian (None, 25, 25, 311)       0
conv2d_4 (Conv2D)            (None, 13, 13, 25)        70000
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 25)          0
batch_normalization_4 (Batch (None, 7, 7, 25)          100
dropout_2 (Dropout)          (None, 7, 7, 25)          0
conv2d_5 (Conv2D)            (None, 3, 3, 25)          5650
max_pooling2d_5 (MaxPooling2 (None, 2, 2, 25)          0
batch_normalization_5 (Batch (None, 2, 2, 25)          100
global_max_pooling2d_2 (Glob (None, 25)                0
dense_2 (Dense)              (None, 1)                 26
=================================================================
Total params: 75,877
Trainable params: 75,777
Non-trainable params: 100
```

Figure 4.14: Architecture of the CNN used in this project. The first two layers are the custom layers designed to produce the recurrence plots and to black out the lower triangle of the pixels in the plots. The dimensions given in the output shape column are (batch size, window length, window length, number of channels). The batch dimension is unknown at initialization thus the None entry. The total number of parameters is dominated by the number of channels.

The network architecture is loosely based on the Visual Geometry Group(VGG) architecture [19] and the model employed in [18]. The recurrence plots produced in the custom layers are handed to the first of two convolutional layers. MaxPooling, batch normalization and dropout layers are applied for regularization while an initial bias and class weights were given to the model to counteract the class imbalance. The models are trained with a batch size of 2048, a learning rate of 0.0001 and an initial epsilon of 14. The Adam optimizer [20] was used and binary cross-entropy as the loss function. The large batch size was chosen to ensure that every batch contains samples of the interlock class. The parameters are summarized in table (4.6).

learning rate	0.0001
batch size	2048
Optimizer	Adam
Loss	binary cross entropy
initial epsilon	14.
number of filters	25
kernel size	(3,3)
stride layer 1	(2,2)
stride layer 2	(3,3)
pooling size	(2,2)
dropout rate	0.15

Table 4.6: Overview of the parameter settings at the time of writing

Different model layouts and parameter choices such as learning rate and batch size were tested. The testing was done using windows of length 25 time steps. This restriction on the window length was imposed by the available computing resources, specifically the memory which did not allow testing a suitably large variety of parameters, specifically window sizes, if all the features are to be utilized in the one plot per feature variation. In order to test the performance of larger window sizes, feature selection had to be performed. In order to do this in a reasonable time frame, tests on the model performance for windows of different placements with respect to the interlock event were performed.

Algorithm (2) describes the recursive feature elimination process. Note that at the time of writing is was suggested that the target value should be used as primary metric to evaluate the model performance and the algorithm has hence been adapted.

Algorithm 2: Recursive feature elimination

```
1  channels = list of available channels ;
2  while number of channels > 0 do
3      for channel in channels do
4          build a model channel ;
5          calculate AUC for that model ;
6      end
7      remove channels with highest n AUC values ;
8      train model with the remaining channels;
9  end
10 Select channel set with the best performance;
```

A recursive feature elimination procedure is employed to identify a minimal set of features for the CNN with recurrence plots model. The procedure was done using windows of length 25 time stamps. The training set contained all interlock samples in the training range, as well as the first and last stable windows in that range and validated on randomly chosen 16% of stable windows in the validation set as well as all interlock instances in the validation set. The selection of stable windows in the training and validation set was necessary for performance reasons.

Using the procedure described in algorithm (2) a set of 101 features was identified. Expert opinion, as well as the random forest feature selection results, suggested that this set could be further reduced. Another round of recursive feature selection was performed starting from the previously selected 101 features. The same method as described above was applied with a step size of 5 instead of 30. This lead to a set of 11 features presented in table (4.10).

4.2.2 Validation of the implementation

In order to check the recurrence plot implementation used here, recreation of the results resented in [18] was attempted. The authors applied their model to data sets from the UCR Time Series Classification Archive which are available as downloads from the archive's website [21].

The authors use time embedded recurrence plots with possible variations in the model parameters for each dataset. These modifications were not disclosed in the paper and could not be communicated upon request. As the given time constraints did not allow a hyper parameter search to find the optimal parameters for every dataset, the reproduction was done using the architecture given in the paper with a learning rate of 0.0001 and a batch size of 64. Time embedding was not used in this reproduction as the parameters needed, embedding dimension and time delay, are unknown. The authors also indicate that the images were scaled to different sizes, depending on the given dataset, which was also not attempted here for the same reason as stated above.

Note on time delay embedding The recurrence plots used in the paper are done using time delay embedding. In this method one has two parameters. The embedding dimension m and the time delay τ. Equation (4.1) shows an example for $m = 4$ and $\tau = 2$.

$$\vec{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{pmatrix} \rightarrow \vec{x}' = \begin{pmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \\ 3 & 5 & 7 & 9 \\ 4 & 6 & 8 & 10 \end{pmatrix} \tag{4.1}$$

The original signal has length $n = 10$. The dimensions of the embedded signal $\vec{x}'$ are then (d_E, m) with $d_E = n - (m-1) * \tau$. One can then construct a recurrence plot from the obtained signal $\vec{x}'$ in the manner of the CNN model presented above.

Results Figure(4.15) compares the obtained recurrence plots with the ones presented in the paper for the respective data sets.

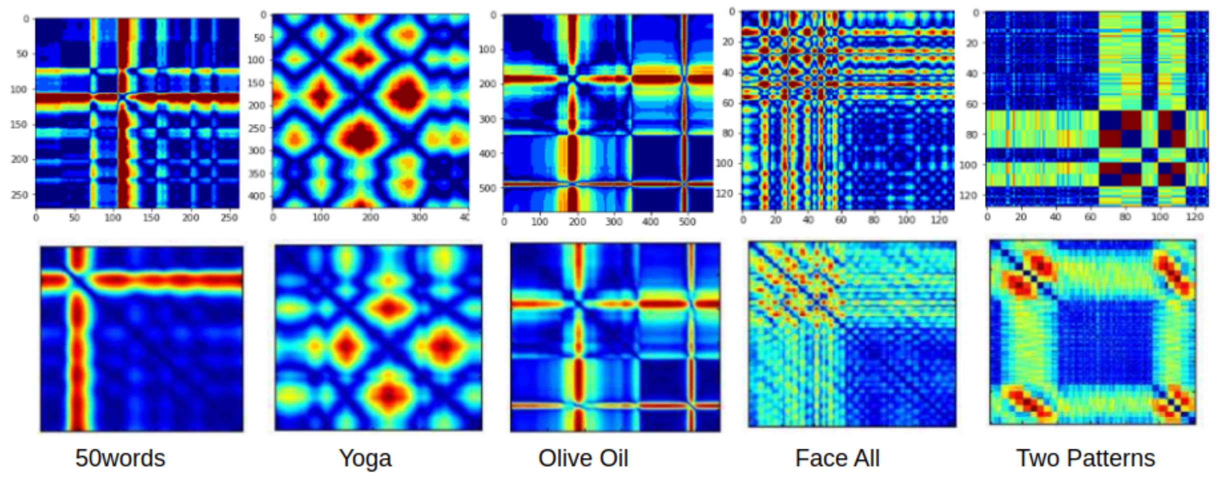


Figure 4.15: Above: Recurrence plots obtained with the RecurrencePlot layer. Below: Recurrence plots presented in the paper.

Most reproductions seem to pick up the same general structures but do differ in the details. The recurrence plot for the Olive Oil sample is the closest to the originalẎoga and Face All are also fairy similar, though with differences in the pattern positioning for the Yoga sample and pattern details in case of the Face All sample. The differences may be attributed to the lack of time embedding and other differences in parameter choices.

Table (4.7) compares the error rates presented in the paper with the results obtained here for all binary UCR data sets used in the paper.

Dataset	**Paper error rate**	**Obtained error rate**	**Confusion Matrix** $\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}$
Coffee	0	0.04	$\begin{bmatrix} 13 & 1 \\ 0 & 13 \end{bmatrix}$
Wafer	0	0.003	$\begin{bmatrix} 5497 & 19 \\ 1 & 646 \end{bmatrix}$
ECG200	0	0.24	$\begin{bmatrix} 53 & 14 \\ 10 & 53 \end{bmatrix}$
GunPoint	0	0.03	$\begin{bmatrix} 72 & 2 \\ 3 & 72 \end{bmatrix}$
Lightning2	0	0.27	$\begin{bmatrix} 27 & 11 \\ 5 & 17 \end{bmatrix}$
Yoga	0	0.28	$\begin{bmatrix} 1144 & 372 \\ 463 & 1020 \end{bmatrix}$

Table 4.7: Only the results for the binary datasets used in the paper were compared.

The published error rate of 0 could not be reached for any of the considered data sets. The results obtained for the data sets Coffee, Wafer and Gunpoint are fairly close to the published error rate with only a few miss-classified samples. The obtained error rates for the other three data sets are all below 0.3 .

Discussion The obtained recurrence plots, as seen in figure (4.15), as well as the computed error rates, presented in table (4.7), are similar enough to their counterparts presented in [18] to suggest that the RecurrencePlot layer does indeed work in the intended manner.

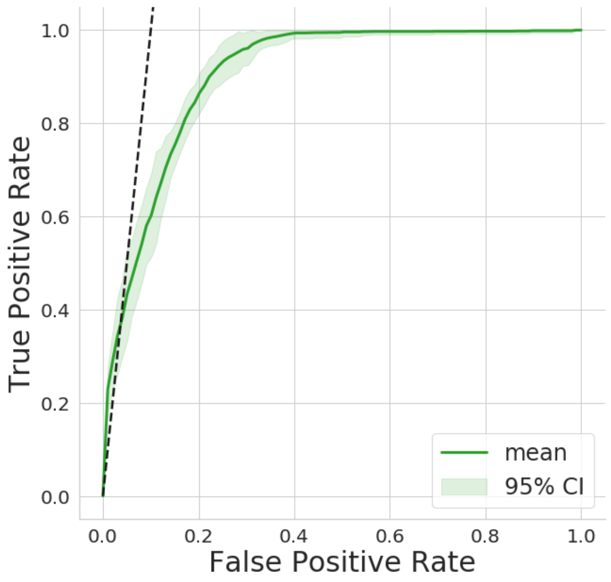
The differences in the obtained results may be attributed to the lack of time embedding and other differences in parameter choices.

4.2.3 Results

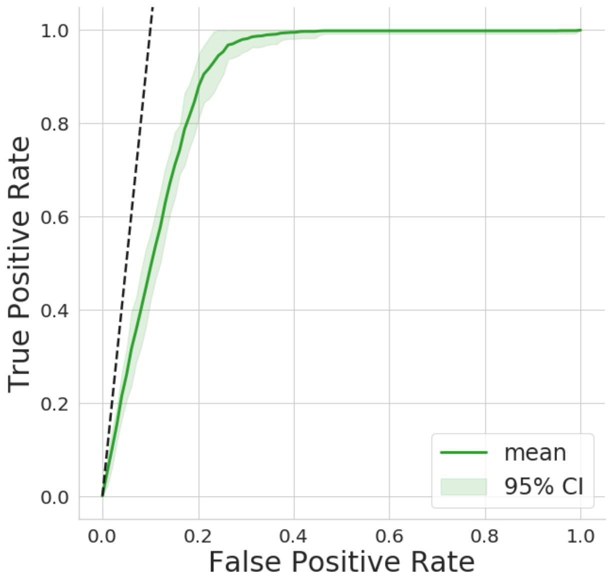
The recursive feature elimination was done using a model with the above described architecture and windows of length 25 timestamps. The model was trained on the last stable windows and validated on randomly chosen 16% stable windows in the validation set. The window size was dictated by performance issues for the one plot variation used with all features. Table (4.8) and figure (4.16) present the performance of the model configuration used for the recursive feature elimination for the one plot and one plot per feature variation.

	One plot per feature	**One plot**
Target value	0.16 ± 0.02	0.003 ± 0.005
AUC	0.905 ± 0.006	0.887 ± 0.009
Beam time lost [s/interlock]	22.4 ± 0.5	24.97 ± 0.06

Table 4.8: Target value and AUC for the ROC curves obtained with models trained on windows of length 25 time stamps for the one plot and one plot per feature variation.



a One plot per feature



b One plot

Figure 4.16: 25 time steps model trained without december data. Left: One plot per feature variation. Right: One plot variation.

Both models seem fairly stable. The one plot per feature variation does achieve a higher target value, as well as a significantly higher AUC value than the one plot variation.

Figure (4.17) and table (4.9) present the results obtained with the one plot and one plot per feature variation using the model trained on window length 25 time stamps for different positions of stable windows with respect to the interlock event.

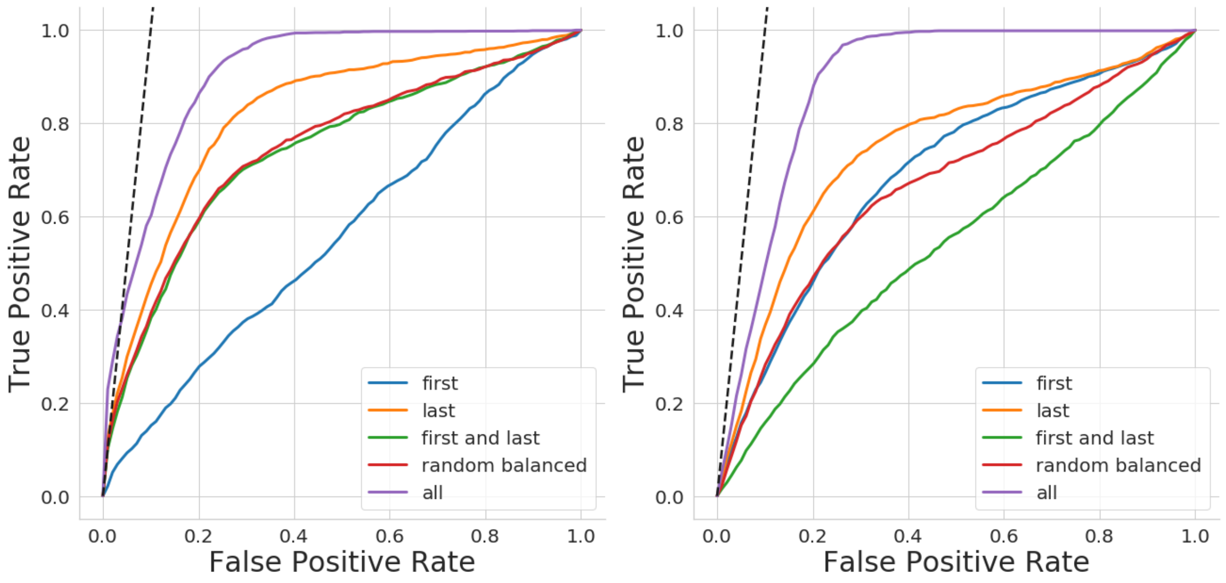


Figure 4.17: Right: One plot per feature. Left: One plot

	First	**Last**	**First and last**	**Random balanced**
Target value one plot per feature	0.0.002 ± 0.004	0.06 ± 0.04	0.05 ± 0.04	0.06 ± 0.05
AUC one plot per feature	0.6 ± 0.2	0.8 ± 0.1	0.7 ± 0.2	0.8 ± 0.2
AUC one plot	0.7 ± 0.2	0.7 ± 0.2	0.5 ± 0.2	0.7 ± 0.2

Table 4.9: Target value and AUC for the ROC curves obtained with models trained on windows of length 25 time stamps on the one plot per feature and the one plot variation. The target value for the one plot variation have been omitted as they are all found to be 0.

In both the one plot per feature and the one plot variation all window positions show very high variability. Thus no conclusion can be drawn as to the performance of the different window positions on the AUC metric. Regarding the distance to target metric, only the one plot per feature variation presents shows positive values, though all variations present high variability here too. No window position for either variation seems to outperform the model trained on all stable windows.

Recursive feature elimination (RFE) Figure (4.18) presents the AUC and target values for the 25 time stamps model trained on the remaining feature sets of each round of the RFE process. 30 features are eliminated after each round.

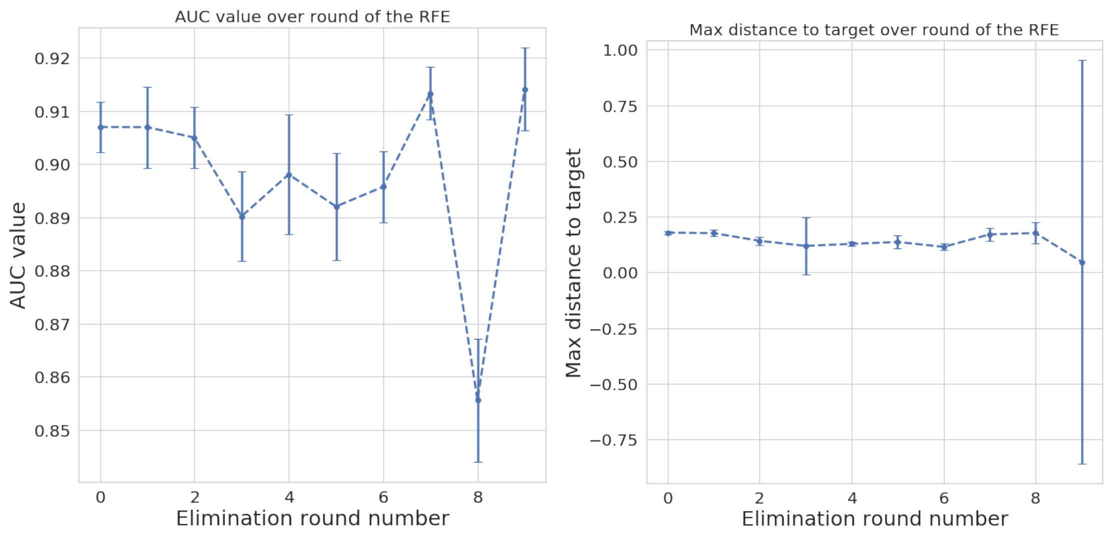


Figure 4.18: RFE results round 1. 25ts model trained and validated on the features in the remainder set for each round of elimination. Three rounds of bootstrapping were applied.

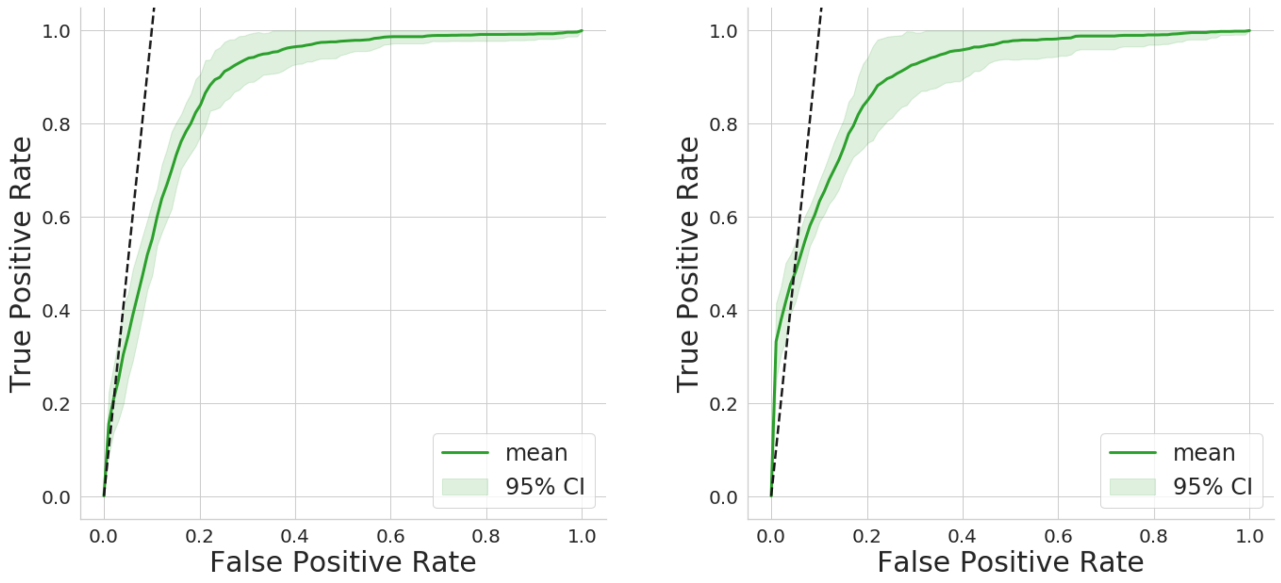
The AUC scores remain fairly stable up until round 8, with only a slight drop at and after round 4 and a slight increase at round 7. The features surviving to round 9 seem to most promising here. The target values remain fairly stable too with a slight increase in value after round 6 and a remarkable increase in variance at round 9. Based on both metrics the features obtained in round 7 seem to be most promising.

The features so obtained were now used as base features for the next round of elimination. Here five channels are discarded at each round. Table (4.10) presents the eleven seemingly most relevant channels found in the recursive feature elimination process detailed above. A list with the top 101 channels from the first round of eliminations can be found in the appendix in table (7.1).

CR1IN:IST:2	CR3OT:IST:2	FMX:IST:2	MHB9O:ILOG:2
CR3DSCO:IST:2	CR5DNO:IST:2	INKOX:ILOG:2	MRTC1WR:IST:2
CR3IN:IST:2	EECF2A:ILOG:2	MHS18:IST1:2	

Table 4.10: Best performing set of features selected from the previously selected 101 by recursive feature elimination with a step size of 5.

Using these 11 channels the model was then further tuned by hand with the window length and amount of sub-sampling as additional parameters. Figure (4.19) presents the ROC curve for a model trained on windows of length 500 time steps, sub-sampled at every second step, trained on the 11 features extracted using the RFE procedure described above.



a One plot b One plot per feature

Figure 4.19: Performance of a model trained on windows of length 500 time stamps using the 11 features selected in the RFE. Left: One plot version. Right: One plot per feature.

	One plot	**One plot per feature**
Target value	0.09 ± 0.04	0.25 ± 0.04
AUC	0.88 ± 0.01	0.90 ± 0.02
Beam time lost [s/interlock]	23.6 ± 0.8	20.5 ± 0.7

Table 4.11: Target value and AUC for the ROC curves obtained with models trained on the one plot and one plot per featue variations.

The ROC curve shows a similar behavior after a FPR of about 0.1 for both the one plot and the one plot per feature variation. Both variations also seem quite stable though this can only be asserted with a higher amount of bootstrapped runs. The one plot per feature variation achieves a higher target value compared to the one plot variation. A distinction between the model can also be observed in the classification of interlock types as shown in figure (4.20). Note that a “interlock-score” threshold of 0.5 was applied here.

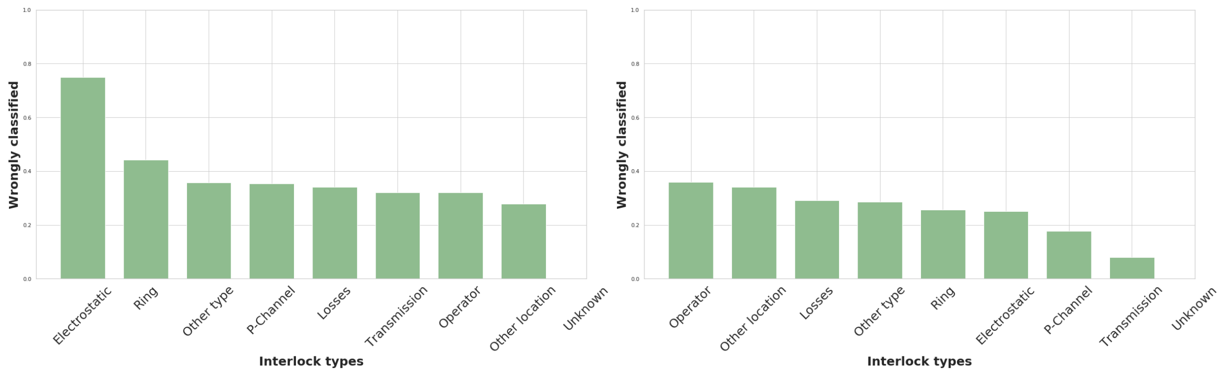
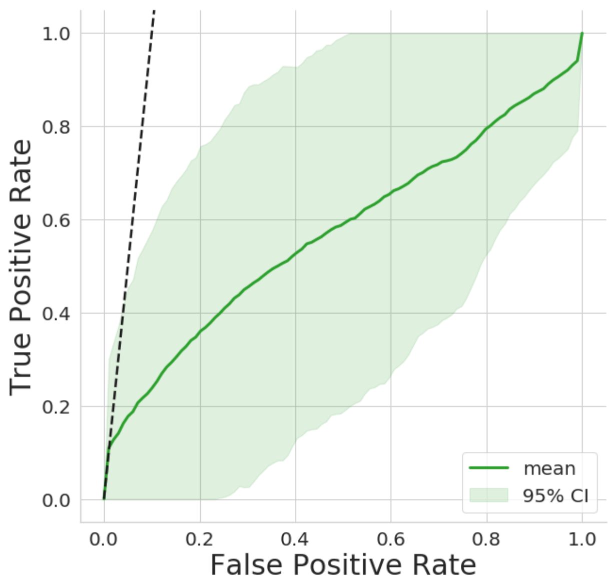


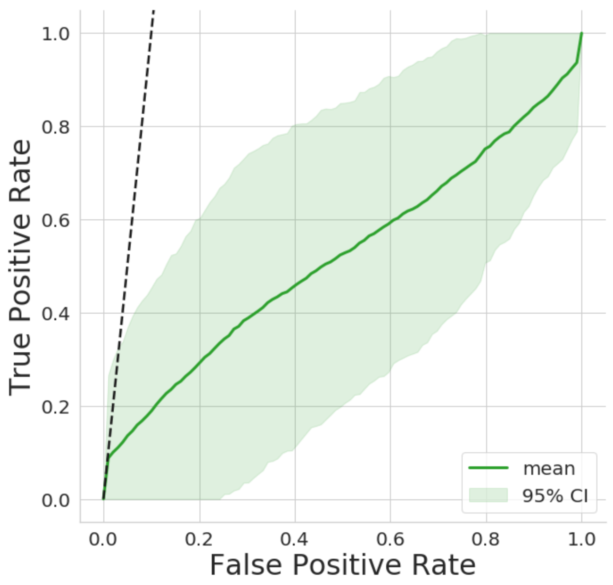
Figure 4.20: Performance of the CNN model trained on windows of length 500 time stamps by interlock type. Left: mode trained using the one plot variation. Right: Model trained using the one plot per feature variation.

The one plot per feature method outperforms the one plot variation for most types except “Operator” and “Other Location”. The difference is especially relevant for the classes “Electrostatic” and “Ring” .

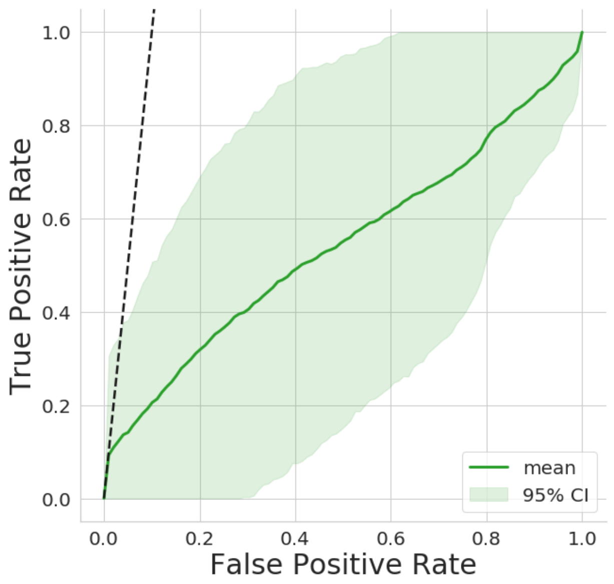
The performance of the model of window length 500 sub-sampled at every second steps was analyzed when trained on stable samples of different positions with respect to the interlock event. The results are displayed in figure (4.21) and table (4.12).



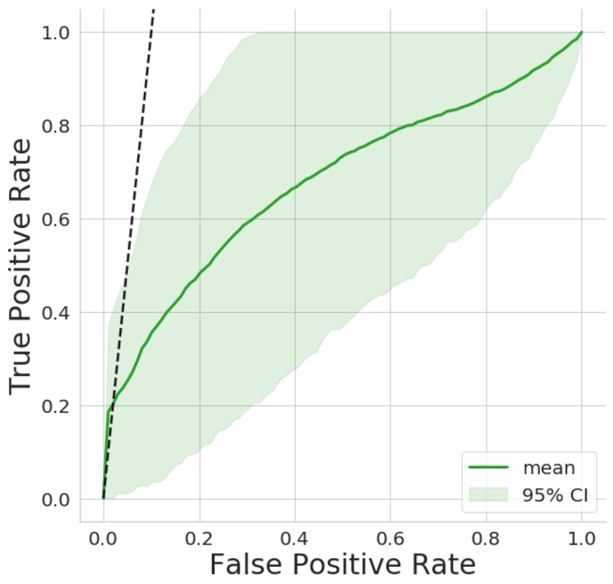
a Random stable windows



b Last stable windows



c First stable windows



d First and Last stable windows

Figure 4.21: CNN model trained on all interlock windows in the training set as well as on stable windows at different position with respect to the interlock windows. All stable windows were used for the validation.

	First	**Last**	**First and last**	**Random balanced**
Target value	0.06 ± 0.08	0.06 ± 0.07	0.14 ± 0.07	0.07 ± 0.07
AUC	0.5 ± 0.2	0.5 ± 0.1	0.7 ± 0.1	0.6 ± 02

Table 4.12: Target values for the ROC curves obtained with models trained on windows of length 500 time stamps sub-sampled at every second time stamp and using the one plot per feature variation.

All windows positions seem quite unstable, and no definite conclusion to their performance can be made. Note that the model trained on the “first and last”stable windows does seem to outperform the other variations in terms of the achieved target value.

Figure (4.22) presents the cross validation results of the 25 time stamps model over different combinations of data splits for shuffled and time ordered data. The ROC curves presented here are obtained over single runs.

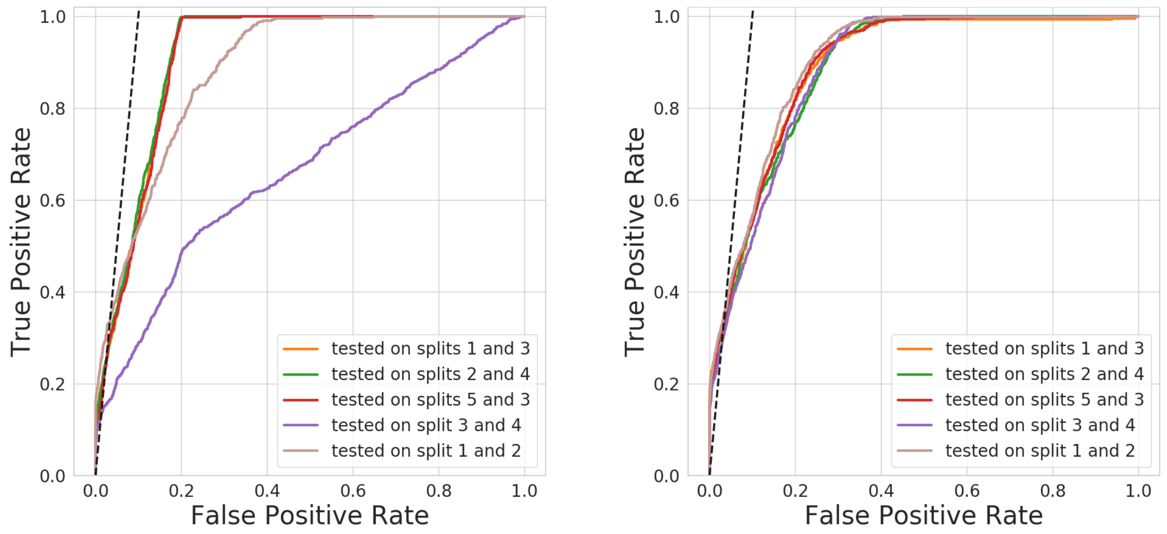


Figure 4.22: Cross validation on different train/test combinations of data splits. In the figure on the right the samples were shuffled before being split.

The model trained on shuffled data shows consistent results over all combinations of splits, the models trained on time ordered data do not. In particular, one observes a drop in performance for models tested on two consecutive splits that is not present for models tested on sets separated by training data.

4.2.4 Discussion

The one plot per feature variation achieves consistently higher target values and is thus preferable for the task of beam time loss reduction.

The difference in behavior of the one plot and the one plot per feature variation in terms of successful interlock classification per type is interesting and may benefit from further investigation.

The selection of stable windows for training the model during the recursive feature elimination is not ideal but the selection was necessary for performance reasons and the behavior of the model with these settings has been evaluated. As the recursive feature elimination is based on comparison with the same data, and same model settings the results should be fine even with the model not at optimal performance. Thus the set of relevant features found by recursive feature elimination may not be optimal. It may also not be unique. As a great number of the channels seem to be highly correlated some or all of the found channels may be replaced by highly correlated alternatives. Further investigation such as a more rigorous feature elimination employing bootstrapping for all intermediary models as well as a smaller elimination step size may be of use, although this would require a substantial amount of time and resources. A focused correlation analysis may be more sensible.

The results obtained for the different positions of stable windows, for the 25 time steps model and more pronounced for the 500 time stamps model, may suggest that not all stable windows are equal for this model. However the high variability of the models does not allow a definite conclusion and further investigation is needed here. A variation of the ten minutes buffer zone around the interlocks may yield interesting results.

The time ordered data presents a higher variability over the different split combinations while the shuffled results show a near identical performance over the splits. This may indicate that the model could struggle to generalize when applied to a new dataset. This needs further investigation. It might also be worth to consider fine-tuning of an existing model if it is to be applied to a new dataset. For instance a model could be trained on data from September then fine-tuned with data from December before being tested on the rest of the December data.

4.3 Model Comparison

The random forest model trained on windows of length 2000 time stamps is compared to the CNN model trained on windows of length 500 time stamps, sub sampled at step length 2.

Figure (4.23) presents the ROC curves of the random forest and CNN models on one plot for better comparison.

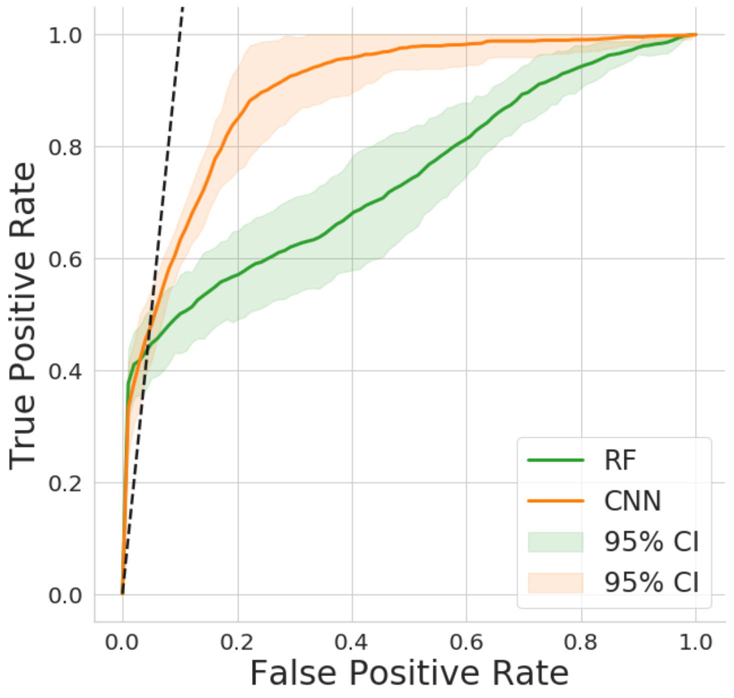


Figure 4.23: Random forest model trained on windows of length 2000ts and CNN model trained on windows of length 500ts sub-sampled at every second time stamp. The random forest model was trained on all features, the CNN model on the selected 11 features.

	Random forest	**CNN**
AUC	0.74 ±0.3	0.90 ± 0.02
Beam time lost [s/interlock]	19.4 ± 0.5	20.5 ± 0.7

Table 4.13: Metrics for the models presented in figure (4.23).

The Random Forest Model achieves a slightly lower beam time loss than the CNN. However considering the ROC plots the CNN has more potential for further improvements.

4.3.1 Selected features

For both the random forest and the CNN with recurrence plots models, the amount of channels necessary to achieve top performance could be significantly reduced. The channels retained by each methods are mostly non-overlapping.

Figure (4.24) and table (4.14) compare the overlap of the features in each selection.

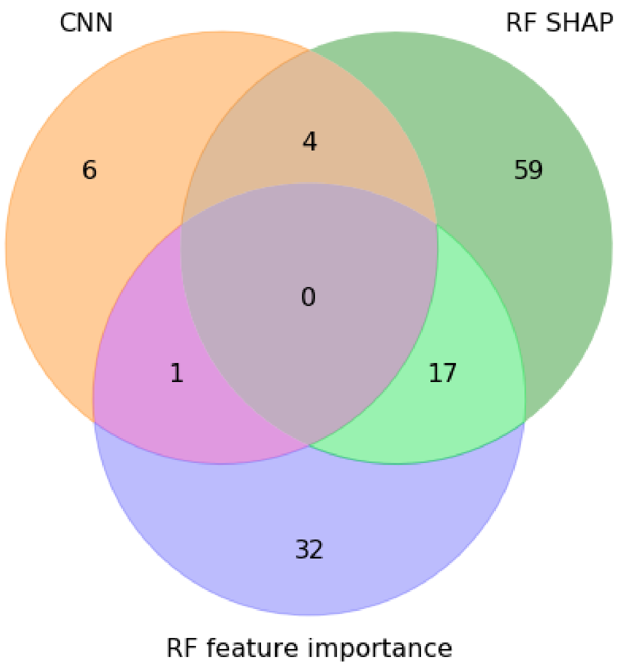


Figure 4.24: Venn diagram illustrating the relationship between the different feature sets. The numbers indicate the numbers of elements in each set. In the SHAP set the top 80 selected features are included and in the feature importance sets the top 50.

CNN and RF feature importance	**RF feature importance and RF SHAP**	**RF SHAP and CNN**
CR5DNO:IST:2	SHD13Y:IST:2	FMX:IST:2
	BHE2LTE:IST:2	CR3OT:IST:2
	BHE1LTE:IST:2	MRTC1WR:IST:2
	BHE1TWA:IST:2	CR1IN:IST:2
	CR5IN:IST:2	
	MHVBON:ALTF:2	
	CR5PN:IST:2	
	BHE4LTE:IST:2	
	CR4OT:IST:2	
	CR5T:IST:2	
	CRPH34:IST:2	
	SHD13Y:IST:2	
	BHE3LTE:IST:2	
	CR5DNU:IST:2	
	BHE3LTA:IST:2	
	BHE2LTA:IST:2	
	BHE1LTA:IST:2	

Table 4.14: Features present in the selections obtained for the different methods and models. Only the intersection features from figure (4.24) are indicated.

The features selected for the CNN model are barely present in the random forest selection. This may be due to the selection methods employed or maybe the different methods do extract different kinds of information from the respective channels. As the performance results of the random forest model and the CNN with respect to the stable window positions are very distinct they do indeed seem to rely on different information, at least for the classification of stable windows.

Figure (4.25) presents the correlations between the top 80 channels found by the SHAP method for the random forest and the top 11 features from the CNN feature selection.

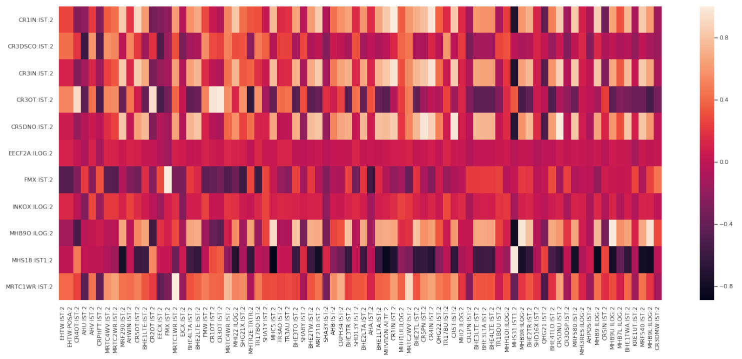


Figure 4.25: Correlations between the top 80 channels found by the SHAP method for the Random Forest and the top 11 features from the CNN feature selection.

Apart from the channels INKOX:ILOG:2 and EECF2A:ILOG:2, all channels present in the CNN selection are either present in the SHAP selection or show a high positive or negative correlation value to some of the channels present in the SHAP selection.

Figure (4.26) presents the correlations between the top 50 channels found by the feature importance method for the random forest and the top 11 features from the CNN feature selection.

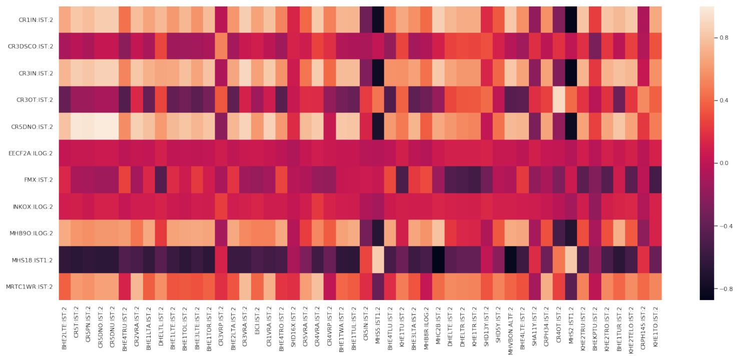


Figure 4.26: Correlations between the top 50 channels found by the feature importance method for the Random Forest and the top 11 features from the CNN feature selection.

Here too apart from the channels INKOX:ILOG:2 and EECF2A:ILOG:2, all channels present in the CNN selection show a high positive or negative correlation to at least one of the channels present in the feature importance selection.

4.3.2 Performance of the selected features on the other model

In order to asses whether the features selected by each method do in principal contain the same information, the performance of the models on the features selected by the other method was evaluated.

Figure (4.27) compares the ROC curves of the random forest model trained on windows of length 1 time stamp using all features to the performance of the same model trained on only the 11 features selected by the CNN model.

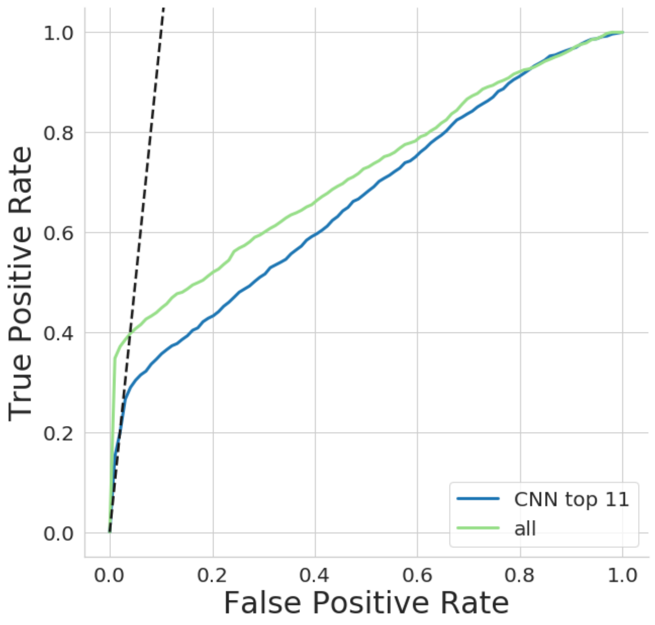


Figure 4.27: Random forest model trained on windows of length 1 using all available features or only the 11 selected for the CNN model.

The features selected by the CNN are clearly not optimal for the random forest model. The achieved performance cannot compete with the one obtained using all available features.

Figure (4.28) compares the performance of the CNN model trained on different sets of the SHAP random forest feature selection.

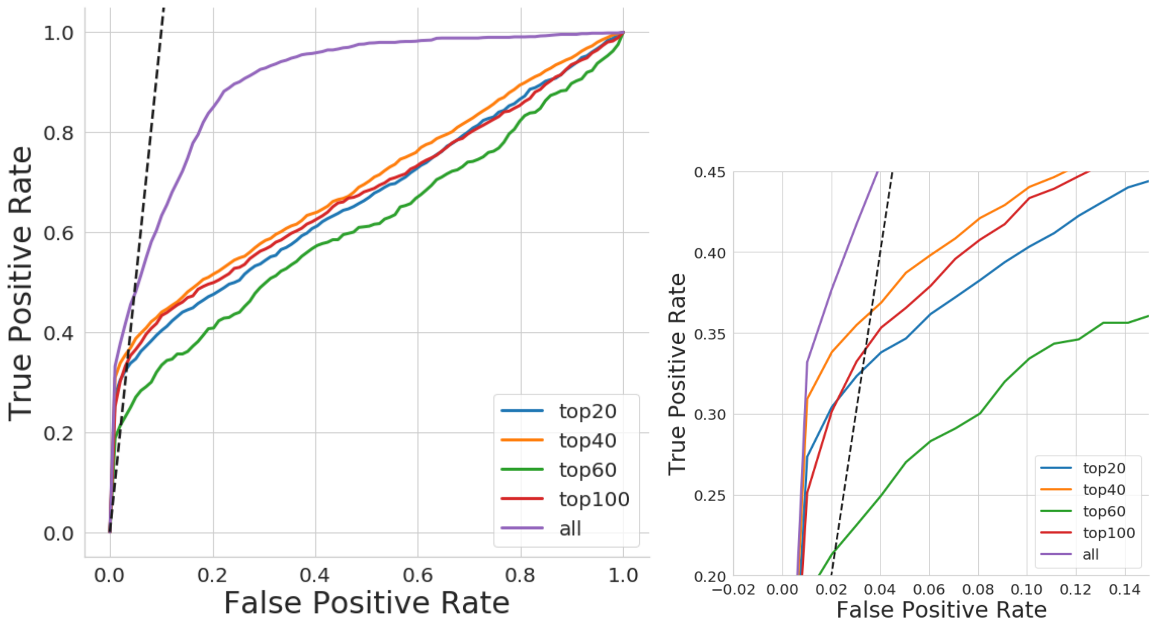


Figure 4.28: CNN model trained on windows of length 500 time stamps, sub-sampled at every second time stamp. The model performance given different sets of the SHAP features obtained for the random forest model was evaluated. The “all ”configuration uses all 311 features.

Note that the model trained on the top 20 features outperforms the model trained on the top 60 features and that the top 40 model outperforms all of the other random forest SHAP feature combinations. Thus adding more features to the set does not seem to necessarily improve the model performance.

4.3.3 Discussion

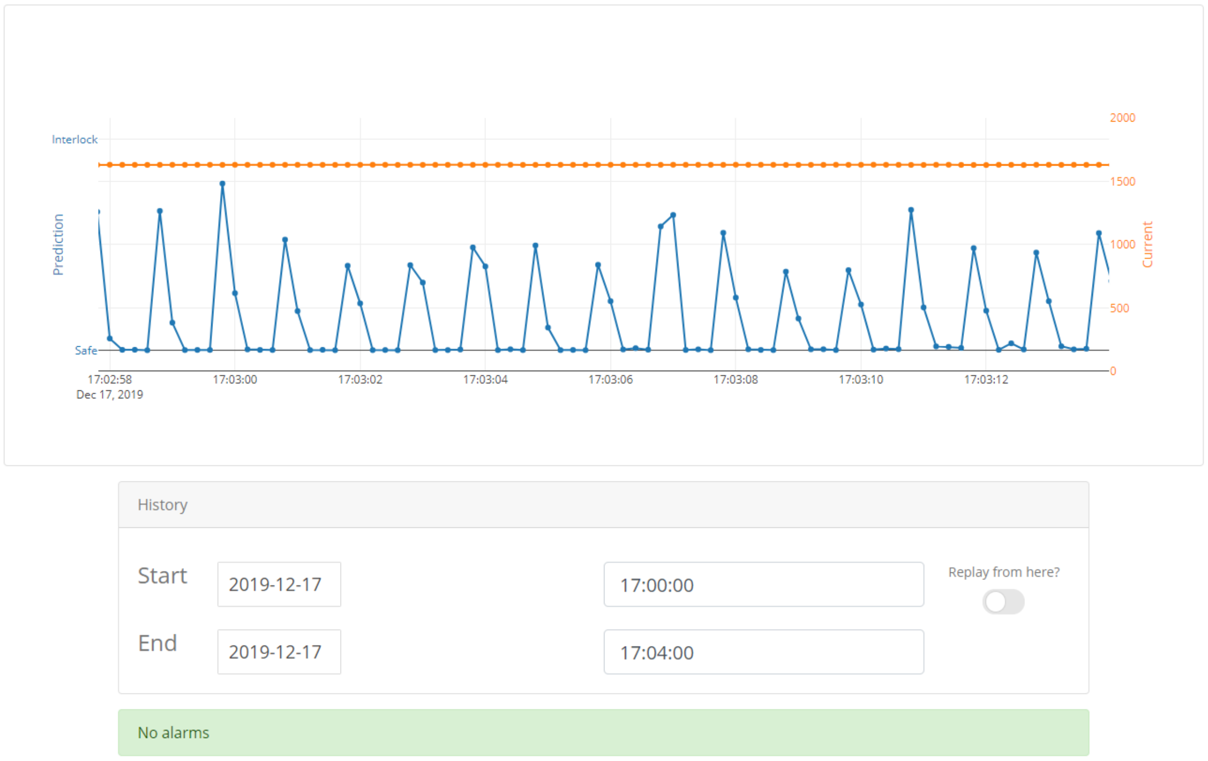
As the random forest model and the CNN model were trained and evaluated on slightly different data sets all one should be careful with drawing general conclusions from the above comparison. The random forest model was applied to the data set with randomized interlock time stamps, while the old interlock time stamp variation was used for the CNN model. While this difference has no impact on the channel data itself, it was found that the CNN model extracted information from the time stamp configuration, see section 5. This may have allowed the CNN model to achieve the observed performance with less features than would otherwise be necessary. One should thus be careful to draw conclusions from the random forest performance on the CNN feature selection. The feature set comparison should be redone as soon as a working CNN model for the adjusted dataset is established.

Chapter 5

Limitations and possible solutions

The following considerations concern the CNN model, as the results presented for the random forest model have been obtained using randomized interlock time stamps.

Shortly before the end of this project, results were obtained that suggest an information leakage to the CNN via the labeling of the interlock time stamps. Evidence for this claim was first observed in the results from the simulated live prediction by Jaime Coello presented in figure (5.1).



History

Start	2019-12-17	17:00:00	Replay from here?
End	2019-12-17	17:04:00	

No alarms

Figure 5.1: Performance on live predictions for a CNN model trained on windows of length 25 time stamps.Figure courtesy of Jaime Coello.

The CNN periodically predicts a high interlock score in a stable region.

Further investigation done by Jochem Snuverink revealed that this periodicity matched the labeling pattern of the interlock timestamps in the dataset. All interlocks have a time stamp of x.0 seconds or x.2 seconds. Interlock events that happen before October 4th have a time stamp of x.2 seconds and all interlocks happening afterwards have a time stamp of x.0 seconds, see figure (5.2).

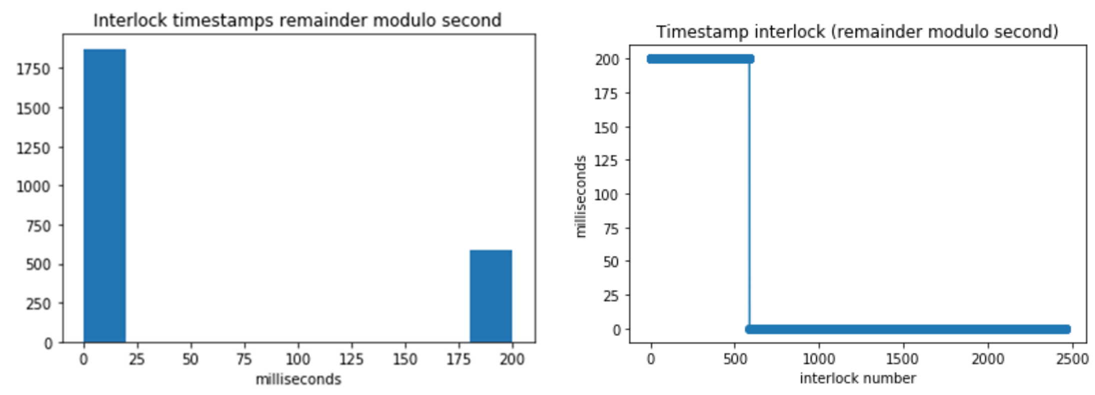


Figure 5.2: Remainder of the interlock time stamps in the 2019 data in milliseconds. All interlocks have a time stamp of x.0 seconds or x.2 seconds. Specifically up until October 4th the interlocks have a time stamp of x.2 and afterwards they have a time stamp of x.0. Figure by Jochem Snuverink

Cross validation of the 25 time stamps CNN model trained and tested on all features shows a relevant drop in performance in the case of testing on the September data and training on all else. The models were trained using only the last stable windows for timing reasons.

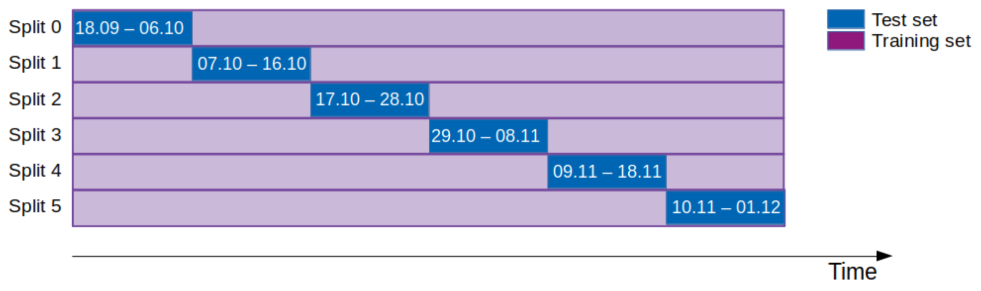


Figure 5.3: Illustration of the data splitting for the 6-fold cross-validation used in figure (5.4).

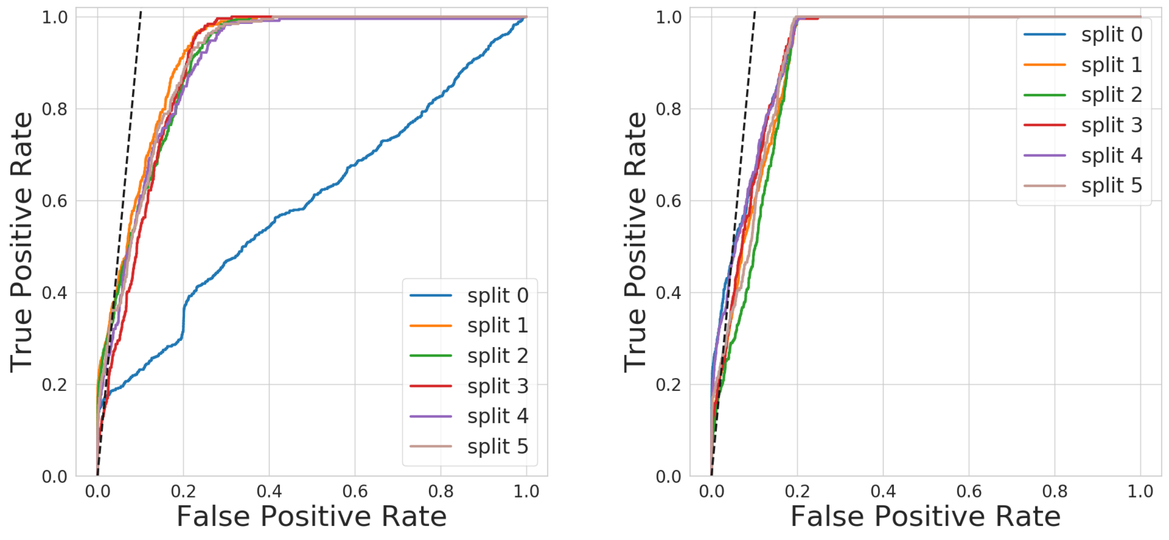


Figure 5.4: Cross validation of the CNN model trained on windows of length 25 time stamps and all available features. Left: all the data has been used. Right: the September data has been omitted.

A more precise investigation of the data included in the 0th split from figure (5.4) presented in figure (5.5) suggest that the performance drop coincides with the change in time step labeling, namely the 4th of October.

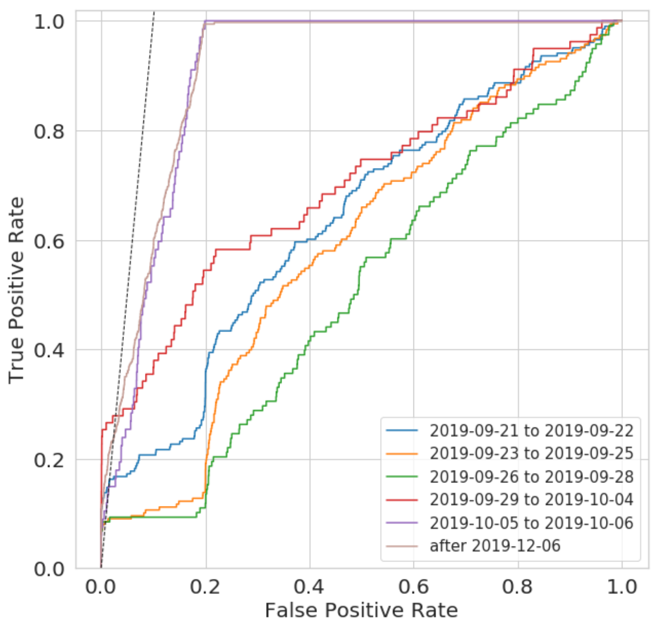


Figure 5.5: ROC curve for the model tested on parts of the September and October data. A test on some of the December data is included as reference.

It thus appears that the CNN model is somehow gaining information about the periodicity of the interlock time stamps and using this to make the predictions. The most likely source of this leakage may lie in a channel that correlates with the time stamps, though no such channel was found at the time of writing.

Solution A solution to this issue was proposed by Jochem Snuverink. The assignment of the interlock time stamp has an uncertainty of about 4 time stamps. It is thus possible to assign a randomized time stamp in that range to each interlock event. Thus we would destroy any periodicity that a model may be able to exploit. A new dataset was build with this configuration and already applied for the random forest model results presented here.

At the time of writing an improvement on the randomization protocol was proposed. The interlock time stamp is to the set equal to the time stamp marking the drop in beam current, channel MHC1:IST:2. A new dataset using this protocol was constructed and will be used henceforth. Time restriction did not permit to reevaluate the random forest model using this interlock time stamp configuration.

Figure (5.6) displays the epoch losses on the training and validation set for the CNN model trained on windows of length 500 time stamps sub-sampled at every second time stamp and using all features on the dataset with randomized interlock time stamps.

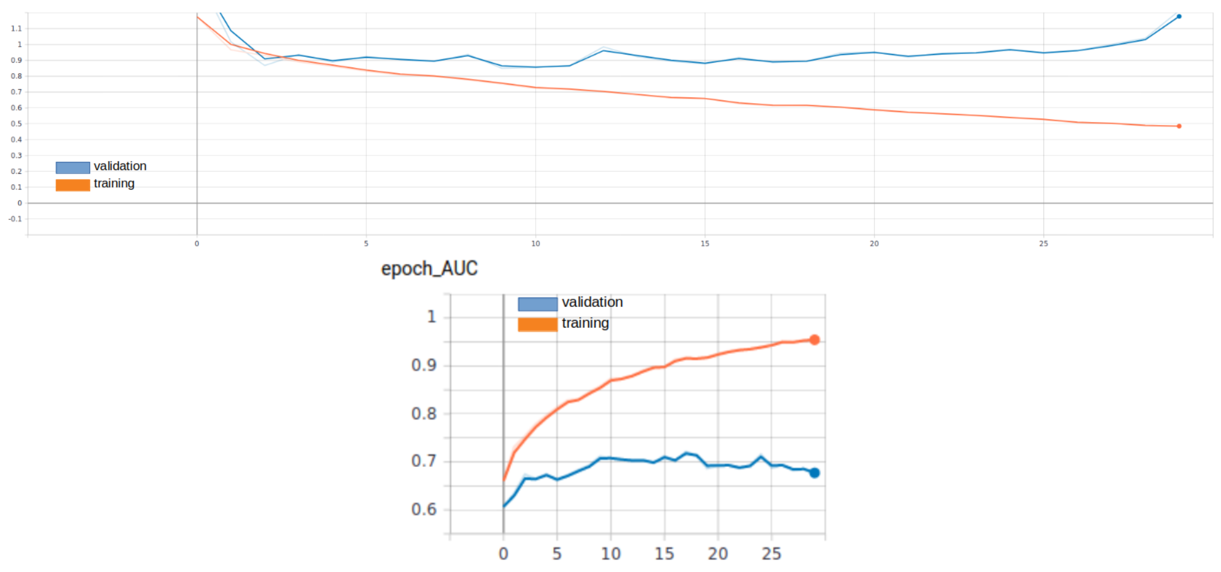


Figure 5.6: Preliminary results for the 500 time stamps model with all features applied to the dataset with randomized interlock time stamps. The batch size had to be reduced to 256 samples for performance reasons.

The batch size for this model had to be decreased from 2048 to 256 for performance reasons. The model does reach a promising loss and AUC value in training but is clearly overfitting. If a new set of relevant features can be selected, this may reduce the overfitting although it may be necessary to alter the network architecture. In any case further investigation is needed to adapt the network to the new dataset with shifted interlock samples.

Chapter 6

Conclusion

The random forest and the CNN with recurrence plots models both reached beam time losses below the non-intervention baseline of 25 seconds per interlock. The random forest trained on windows of length 2000 time stamps (8min) reaches a beam time loss 19.4 ± 0.5 seconds per interlock with respect to the baseline loss of 25 seconds per interlock. The CNN with recurrence plots reaches a beam time loss of 20.5 ± 0.7 seconds per interlock for windows of length 500 time stamps (100s) sampled at every second time stamp using the one plot per feature variation. Although the target values reached by the random forest model are higher, the performance of the CNN model suggests greater potential for further improvements.

Due to the discovery of information leakage by the interlock time stamp labeling the CNN model has to be reevaluated. New protocols to set the interlock time stamps have been introduced. The dataset with randomization of the interlock time stamps, in the manner described in chapter 5, has already been applied for the random forest model. Preliminary results for the CNN model applied to the dataset with interlock time stamps shifted to the beam current drop time stamp are encouraging but need further investigation.

This work is, to our knowledge, the first application of the CNN with recurrence plots model utilizing multivariate time series and a custom neural network layer that allows for trainable hyper-parameters of the recurrence plots. The viability of the method for interlock predictions has been demonstrated although further research is needed.

The late reaction of the random forest model in live predictions suggests that it may not be useful to reduce beam time loss. Simulated live prediction assessment of the CNN model revealed a potential information leakage through the labeling of the interlock timestamps. Randomized interlock time stamps were introduced as a solution. At the time of writing the CNN model has to be readjusted to the new dataset. The preliminary training results indicate that the previously obtained performance may also be attainable with the adjusted network.

It was suspected that a significant number of the channels used in this project are redundant. As the random forest model and the CNN model achieved their respective top performances with a significantly reduced feature set, this suspicion was confirmed.

Chapter 7

Outlook

The next step should be the adaptation of the CNN model to the data using shifted interlock time stamps. As there was not sufficient time to perform a grid search over parameters of the CNN model, this too should be done as a configuration reaching a better beam time lost may still be found.

Due to the time constraint a range of variations to the methods described above could not be explored. Some of them may be of interest in the future. The interlock events are considered here as a single class though they belong to a range of different types. Based on the interlock type classification results form both models, it may be beneficial to reformulate the problem as a multi class classification where each interlock type is assigned an individual label. Other variations of recurrence plots could also be explored. Time embedding like it was used in [18] might be a possibility, discrete recurrence plots or a variation with an individual ϵ_i parameter for each channel i might also be interesting alternatives.

Another approach to explore might be a CNN model designed as a collection of sub models that use windows of different lengths in order to catch effects on different time scales.

The behavior of the random forest model during live testing did not conform to the predictions during testing. It thus might be worth to consider training and testing the model on rolling windows from the start instead of the consecutive non-overlapping windows used so far. This might give a more realistic estimate of the model performance for the end goal of live predictions.

Using flattened recurrence plots as input samples for the random forest model might be another interesting alternative.

Bibliography

[1] T. Fawcett, "Introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, pp. 861–874, 06 2006.

[2] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[3] T. Ganegedara, "Intuitive guide to understanding decision trees."

[4] C. J. S. R. O. Leo Breiman, Jerome Friedman, *Classification and Regression Trees.* Chapman and Hall/CRC, 1984.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016. http://www.deeplearningbook.org.

[6] https://de.wikipedia.org/wiki/Datei:2D_Convolution_Animation.gif.

[7] https://de.wikipedia.org/wiki/Datei:Max_pooling.png.

[8] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 4765–4774, Curran Associates, Inc., 2017.

[9] S. Lundberg, "A unified approach to interpreting model predictions - nips 2017."

[10] S. M. Lundberg, B. Nair, M. S. Vavilala, M. Horibe, M. J. Eisses, T. Adams, D. E. Liston, D. K.-W. Low, S.-F. Newman, J. Kim, *et al.*, "Explainable machine-learning predictions for the prevention of hypoxaemia during surgery," *Nature Biomedical Engineering*, vol. 2, no. 10, p. 749, 2018.

[11] S. M. Lundberg, G. G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S. Lee, "Explainable AI for trees: From local explanations to global understanding," *CoRR*, vol. abs/1905.04610, 2019.

[12] J.-P. Eckmann, S. O. Kamphorst, and D. Ruelle, "Recurrence plots of dynamical systems," *Europhysics Letters (EPL)*, vol. 4, pp. 973–977, nov 1987.

[13] C. L. Webber Jr and J. P. Zbilut, "Recurrence quantification analysis of nonlinear dynamical systems," *Tutorials in contemporary nonlinear methods for the behavioral sciences*, vol. 94, no. 2005, pp. 26–94, 2005.

[14] M. T. J. K. N. Marwan, M. C. Romano, "Recurrence plots for the analysis of complex systems, physics reports, 438(5-6), 237-329, 2007.." www.recurrence-plot.tk. Accessed: 2020-04-01.

[15] https://gitlab.psi.ch/adelmann/mllib. Accessed: 2020-04-01.

[16] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

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

[18] N. Hatami, Y. Gavet, and J. Debayle, "Classification of time-series images using deep convolutional neural networks," 10 2017.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[21] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML, "The ucr time series classification archive," October 2018. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

[22] "Hipa strahlwegbersicht." https://ecm.psi.ch/alfresco/webdav/Projects/HIPA/Layout/01.0.458G_20131216.pdf.

Appendix

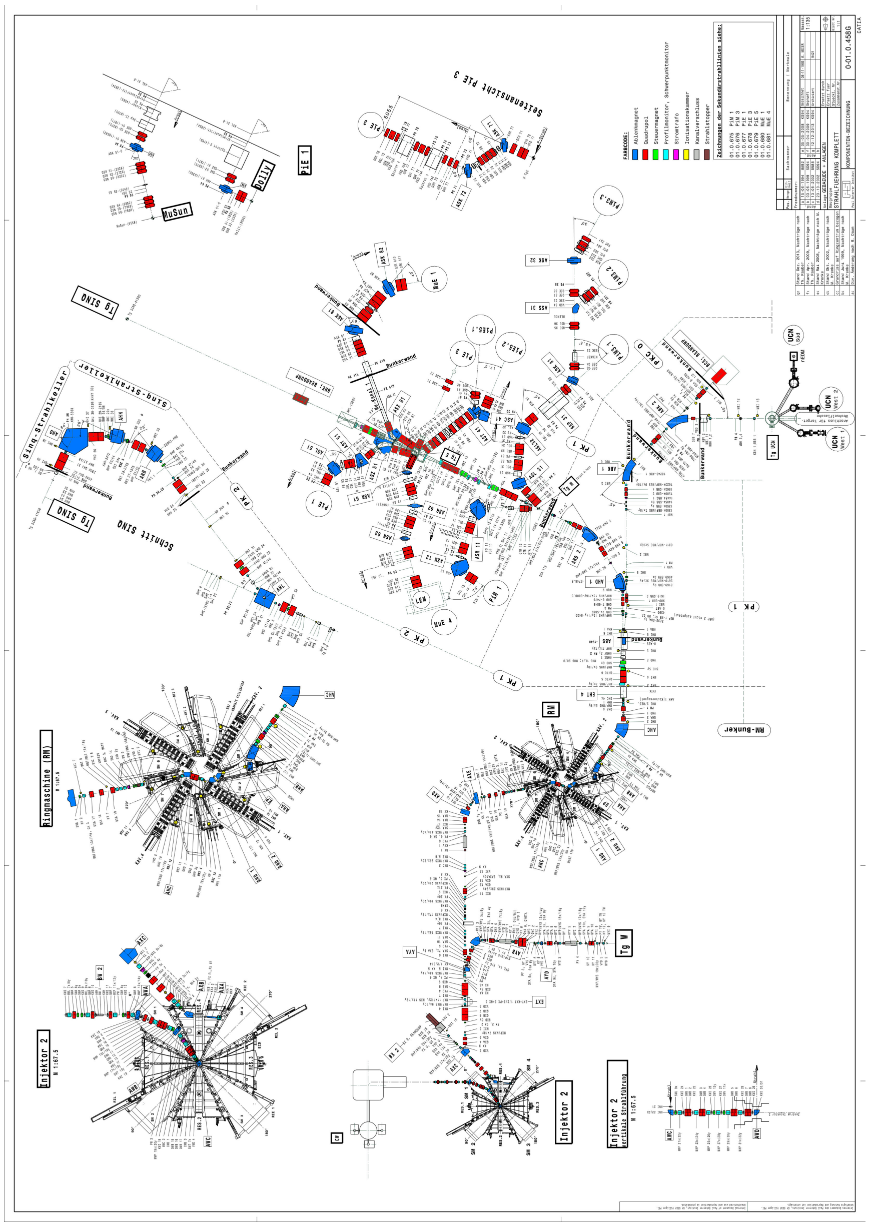


Figure 7.1: Overview of the HIPA facilities with sensor names and locations[22]

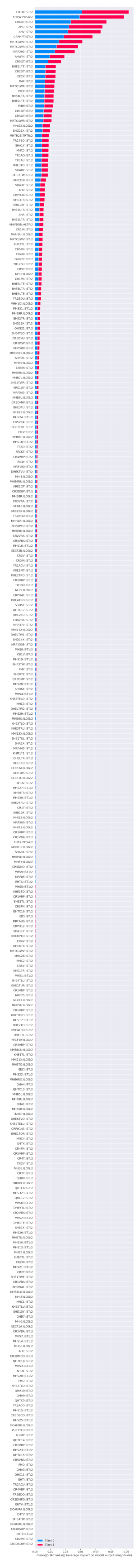
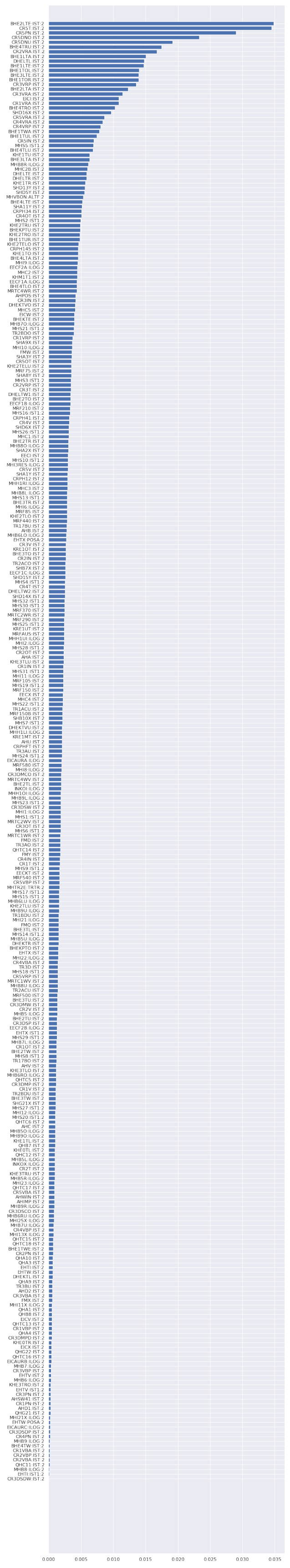



Figure 7.2: The complete rankings obtained by the SHAP (left) and feature importance (right) methods for the random forest model.

AHA:IST:2	CR3DMW:IST:2	DHEKTVO:IST:2	FMX:IST:2	MHB7L:ILOG:2	MHS14:IST1:2	MRTC1WR:IST:2
BHE1TUR:IST:2	CR3DSCO:IST:2	DHEKTVU:IST:2	INKOX:ILOG:2	MHB8:ILOG:2	MHS15:IST1:2	MRTC2WR:IST:2
BHE1TWE:IST:2	CR3DSDW:IST:2	DHELTE:IST:2	KHE0TL:IST:2	MHB8U:ILOG:2	MHS16:IST1:2	QHA10:IST:2
BHE2LTE:IST:2	CR3IN:IST:2	DHELTW2:IST:2	KHE0TR:IST:2	MHB9:ILOG:2	MHS18:IST1:2	QHTC16:IST:2
BHE2TO:IST:2	CR3OT:IST:2	EECF1A:ILOG:2	KHE1TO:IST:2	MHB9O:ILOG:2	MHS2:IST1:2	QHTC17:IST:2
BHE2TU:IST:2	CR3PN:IST:2	EECF2A:ILOG:2	KHE2TELO:IST:2	MHI10:ILOG:2	MHS20:IST1:2	SHB10X:IST:2
BHE3TU:IST:2	CR3T:IST:2	EECF2B:ILOG:2	KHE2TELU:IST:2	MHI13X:ILOG:2	MHS25:IST1:2	SHD16X:IST:2
BHE4TLU:IST:2	CR4IN:IST:2	EECI:IST:2	KHE2TLO:IST:2	MHI2:ILOG:2	MHS26:IST1:2	TR1BDU:IST:2
CR1IN:IST:2	CR4OT:IST:2	EHTW:IST:2	KHE2TRO:IST:2	MHI21:ILOG:2	MHS30:IST1:2	TR2BDO:IST:2
CR1VBP:IST:2	CR4T:IST:2	EHTX:POSA:2	KHE3TRO:IST:2	MHI22:ILOG:2	MHS4:IST1:2	TR2BDU:IST:2
CR1VRA:IST:2	CR4VBP:IST:2	EICI:IST:2	KHE3TRU:IST:2	MHI25X:ILOG:2	MRF105:IST:2	TR3D:IST:2
CR1VRA:IST:2	CR5DNO:IST:2	EICV:IST:2	KRE1MT:IST:2	MHI3RES:ILOG:2	MRF210:IST:2	
CR2VBP:IST:2	CR5PN:IST:2	EICW:IST:2	MHB5O:ILOG:2	MHI8:ILOG:2	MRF290:IST:2	
CR3DMCO:IST:2	CRPH12:IST:2	FMQ:IST:2	MHB5R:ILOG:2	MHI9:ILOG:2	MRF440:IST:2	
CR3DMP:IST:2	CRPH145:IST:2	FMW:IST:2	MHB6RO:ILOG:2	MHS10:IST1:2	MRF540:IST:2	

Table 7.1: Best performing feature set obtained per recursive feature elimination with step size 30.

AHA:IST:2	BHE3TW:IST:2	CR3DSP:IST:2	CRREV:IST:2	EICI:IST:2	KHN31TL:IST:2	MBC1B:IST:2	MHB8L:ILOG:2	MHI31:ILOG:2	MHS26:IST1:2	QHA1:IST:2	SHB10X:IST:2	TR2ACO:IST:2
AHB:IST:2	BHE4LTA:IST:2	CR3DSW:IST:2	DHEKTL:IST:2	EICV:IST:2	KHN31TO:IST:2	MHB1L:ILOG:2	MHB8O:ILOG:2	MHI32:ILOG:2	MHS27:IST1:2	QHA10:IST:2	SHB7X:IST:2	TR2ACU:IST:2
AHC:IST:2	BHE4LTE:IST:2	CR3IN:IST:2	DHEKTR:IST:2	EICW:IST:2	KHN31TR:IST:2	MHB1R:ILOG:2	MHB8R:ILOG:2	MHI33B:ILOG:2	MHS28:IST1:2	QHA2:IST:2	SHD13Y:IST:2	TR2BDO:IST:2
AHD1:IST:2	BHE4TLO:IST:2	CR3OT:IST:2	DHEKTVO:IST:2	EICX:IST:2	KHN31TU:IST:2	MHB21O:ILOG:2	MHB8U:ILOG:2	MHI34:ILOG:2	MHS29:IST1:2	QHA3:IST:2	SHD14X:IST:2	TR2BDU:IST:2
AHD2:IST:2	BHE4TLU:IST:2	CR3PN:IST:2	DHEKTVU:IST:2	FMD:IST:2	KHN33TL:IST:2	MHB21U:ILOG:2	MHB9:ILOG:2	MHI34B:ILOG:2	MHS3:IST1:2	QHA4:IST:2	SHD15Y:IST:2	TR3AO:IST:2
AHIMP:IST:2	BHE4TRO:IST:2	CR3T:IST:2	DHELTE:IST:2	FMEIL:ILOG:2	KHN33TO:IST:2	MHB22L:ILOG:2	MHB9L:ILOG:2	MHI35:ILOG:2	MHS30:IST1:2	QHA9:IST:2	SHD16X:IST:2	TR3AU:IST:2
AHL:IST:2	BHE4TRU:IST:2	CR3V:IST:2	DHELTL:IST:2	FMEIO:ILOG:2	KHN33TR:IST:2	MHB22R:ILOG:2	MHB9O:ILOG:2	MHI35B:ILOG:2	MHS31:IST1:2	QHB7:IST:2	SHD5Y:IST:2	TR3BO:IST:2
AHN:IST:2	BHE4TW:IST:2	CR3VBA:IST:2	DHELTR:IST:2	FMEIR:ILOG:2	KHN33TU:IST:2	MHB28:IBS1:2	MHB9R:ILOG:2	MHI36:ILOG:2	MHS32:IST1:2	QHB8:IST:2	SHD6X:IST:2	TR3BU:IST:2
AHPOS:IST:2	BHEKPTO:IST:2	CR3VBP:IST:2	DHELTW1:IST:2	FMEIU:ILOG:2	KHNX1TL:IST:2	MHB28:IBS2:2	MHB9U:ILOG:2	MHI36B:ILOG:2	MHS4:IST1:2	QHC11:IST:2	SHG21X:IST:2	TR3C:IST:2
AHSW41:IST:2	BHEKPTU:IST:2	CR3VRA:IST:2	DHELTW2:IST:2	FMQ:IST:2	KHNX1TR:IST:2	MHB28:IBS3:2	MHC1:IST:2	MHI37:ILOG:2	MHS5:IST1:2	QHC12:IST:2	SHG22X:IST:2	TR3D:IST:2
AHU:IST:2	BHEKTA:IST:2	CR3VRP:IST:2	EECF1A:ILOG:2	FMW:IST:2	KHNX22L:IST:2	MHB28:POSYI:2	MHC2:IST:2	MHI38:ILOG:2	MHS6:IST1:2	QHG21:IST:2	SHI23X:IST:2	TR4:IST:2
AHV:IST:2	BHEKTE:IST:2	CR4IN:IST:2	EECF1B:ILOG:2	FMX:IST:2	KHNX22R:IST:2	MHB28:IBS4:2	MHC2B:IST:2	MHI38B:ILOG:2	MHS7:IST1:2	QHG22:IST:2	SHI24X:IST:2	TR5:IST:2
AHWIN:IST:2	CR1IN:IST:2	CR4OT:IST:2	EECF1C:ILOG:2	FMY:IST:2	KHNX22TL:IST:2	MHB34L:ILOG:2	MHC3:IST:2	MHI39:ILOG:2	MHS8:IST1:2	QHG23:IST:2	SHI25X:IST:2	TR6:IST:2
ANWIKO:ILOG:2	CR1OT:IST:2	CR4PN:IST:2	EECF2A:ILOG:2	INKOI:ILOG:2	KHNX22TR:IST:2	MHB34O:ILOG:2	MHC4:IST:2	MHI39B:ILOG:2	MHS9:IST1:2	QHG24:IST:2	TR10:IST:2	TR7:IST:2
BHE1LTA:IST:2	CR1PN:IST:2	CR4T:IST:2	EECF2B:ILOG:2	INKOX:ILOG:2	KHNY21O:IST:2	MHB34R:ILOG:2	MHC5:IST:2	MHI3RES:ILOG:2	MHTR2E:TRTR:2	QHI25:IST:2	TR11:IST:2	TR8:IST:2
BHE1LTE:IST:2	CR1T:IST:2	CR4V:IST:2	EECF2C:ILOG:2	KHE0TL:IST:2	KHNY21TO:IST:2	MHB34U:ILOG:2	MHC6:IST:2	MHI4:ILOG:2	MHVBON:ALTF:2	QHI26:IST:2	TR12:IST:2	TR9:IST:2
BHE1TOL:IST:2	CR1V:IST:2	CR4VBA:IST:2	EECI:IST:2	KHE0TR:IST:2	KHNY21TU:IST:2	MHB4L:ILOG:2	MHC6B:IST:2	MHI5:ILOG:2	MRF105:IST:2	QHI27:IST:2	TR13:IST:2	WIKO:ILOG:2
BHE1TOR:IST:2	CR1VBA:IST:2	CR4VBP:IST:2	EECK:ILOG:2	KHE1TL:IST:2	KHNY21U:IST:2	MHB4O:ILOG:2	MHH1LI:ILOG:2	MHI6:ILOG:2	MRF150:IST:2	QHI28:IST:2	TR14:IST:2	
BHE1TUL:IST:2	CR1VBP:IST:2	CR4VRA:IST:2	EECKT:IST:2	KHE1TO:IST:2	KHNY2TO:IST:2	MHB4R:ILOG:2	MHH1OI:ILOG:2	MHI8:ILOG:2	MRF150B:IST:2	QHI29:IST:2	TR15AB:IST:2	
BHE1TUR:IST:2	CR1VRA:IST:2	CR4VRP:IST:2	EECV:IST:2	KHE1TR:IST:2	KHNY2TU:IST:2	MHB4U:ILOG:2	MHH1RI:ILOG:2	MHI9:ILOG:2	MRF210:IST:2	QHJ30:IST:2	TR15CD:IST:2	
BHE1TWA:IST:2	CR1VRP:IST:2	CR5DNO:IST:2	EECW:IST:2	KHE1TU:IST:2	KHNY30O:POSA:2	MHB5:ILOG:2	MHH1UI:ILOG:2	MHS1:IST1:2	MRF290:IST:2	QHJ31:IST:2	TR16AB:IST:2	
BHE1TWE:IST:2	CR2IN:IST:2	CR5DNU:IST:2	EECX:IST:2	KHE2TELO:IST:2	KHNY30O:ILOG:2	MHB5L:ILOG:2	MHI1:ILOG:2	MHS10:IST1:2	MRF370:IST:2	QHJ32:IST:2	TR16CD:IST:2	
BHE2LTA:IST:2	CR2OT:IST:2	CR5IN:IST:2	EHTI:IST:2	KHE2TELU:IST:2	KHNY30TO:IST:2	MHB5O:ILOG:2	MHI10:ILOG:2	MHS13:IST1:2	MRF440:IST:2	QHTC13:IST:2	TR17AO:IST:2	
BHE2LTE:IST:2	CR2PN:IST:2	CR5OT:IST:2	EHTI:IST1:2	KHE2TLO:IST:2	KHNY30TU:IST:2	MHB5R:ILOG:2	MHI11:ILOG:2	MHS14:IST1:2	MRF500:IST:2	QHTC14:IST:2	TR17AU:IST:2	
BHE2TL:IST:2	CR2T:IST:2	CR5PN:IST:2	EHTKTL:IST:2	KHE2TLU:IST:2	KHNY30U:ILOG:2	MHB5U:ILOG:2	MHI11X:ILOG:2	MHS15:IST1:2	MRF540:IST:2	QHTC15:IST:2	TR17BO:IST:2	
BHE2TO:IST:2	CR2V:IST:2	CR5T:IST:2	EHTKTR:IST:2	KHE2TRO:IST:2	KHNY30U:POSA:2	MHB6:ILOG:2	MHI12:ILOG:2	MHS16:IST1:2	MRF580:IST:2	QHTC16:IST:2	TR17BU:IST:2	
BHE2TR:IST:2	CR2VBA:IST:2	CR5V:IST:2	EHTV:IST:2	KHE2TRU:IST:2	KRE1MT:IST:2	MHB6LO:ILOG:2	MHI13X:ILOG:2	MHS17:IST1:2	MRF75:IST:2	QHTC17:IST:2	TR17C:IST:2	
BHE2TU:IST:2	CR2VBP:IST:2	CR5VBA:IST:2	EHTV:IST1:2	KHE3TLO:IST:2	KRE1OT:IST:2	MHB6LU:ILOG:2	MHI16:ILOG:2	MHS18:IST1:2	MRF85:IST:2	QHTC18:IST:2	TR17D:IST:2	
BHE2TW:IST:2	CR2VRA:IST:2	CR5VBP:IST:2	EHTW:IST:2	KHE3TLU:IST:2	KRE1UT:IST:2	MHB6RO:ILOG:2	MHI2:ILOG:2	MHS19:IST1:2	MRFAUS:IST:2	QHTC5:IST:2	TR18ACO:IST:2	
BHE3LTA:IST:2	CR2VRP:IST:2	CR5VRA:IST:2	EHTW:POSA:2	KHE3TRO:IST:2	M3TBPS-L:IST:2	MHB6RU:ILOG:2	MHI21:ILOG:2	MHS2:IST1:2	MRJAHA1:ILOG:2	QHTC6:IST:2	TR18ACU:IST:2	
BHE3LTE:IST:2	CR3DMCO:IST:2	CR5VRP:IST:2	EHTX:IST1:2	KHE3TRU:IST:2	M3TBPS-O:IST:2	MHB7:ILOG:2	MHI21X:ILOG:2	MHS20:IST1:2	MRTC1WR:IST:2	SHA11Y:IST:2	TR18BDO:IST:2	
BHE3TL:IST:2	CR3DMP:IST:2	CRPH12:IST:2	EHTX:POSA:2	KHM1T1:IST:2	M3TBPS-R:IST:2	MHB7L:ILOG:2	MHI22:ILOG:2	MHS21:IST1:2	MRTC1WV:IST:2	SHA1Y:IST:2	TR18BDU:IST:2	
BHE3TO:IST:2	CR3DMPD:IST:2	CRPH145:IST:2	EHTX:IST:2	KHM1T2:IST:2	M3TBPS-U:IST:2	MHB7O:ILOG:2	MHI23:ILOG:2	MHS22:IST1:2	MRTC2WR:IST:2	SHA2X:IST:2	TR1ACO:IST:2	
BHE3TR:IST:2	CR3DMW:IST:2	CRPH23:IST:2	EICAURA:ILOG:2	KHM2T1:IST:2	M3TBPS-Z12:SOL:2	MHB7R:ILOG:2	MHI24:ILOG:2	MHS23:IST1:2	MRTC2WV:IST:2	SHA3Y:IST:2	TR1ACU:IST:2	
BHE3TU:IST:2	CR3DSDP:IST:2	CRPH34:IST:2	EICAURB:ILOG:2	KHMEI:ILOG:2	M3TBPS-Z14:SOL:2	MHB7U:ILOG:2	MHI25X:ILOG:2	MHS24:IST1:2	MRTC4WR:IST:2	SHA8Y:IST:2	TR1BDO:IST:2	
CR3DSCO:IST:2	CR3DSDW:IST:2	CRPH41:IST:2	EICAURC:ILOG:2	KHMET1:IST:2	MBC1:IST:2	MHB8:ILOG:2	MHI3:ILOG:2	MHS25:IST1:2	MRTC4WV:IST:2	SHA9X:IST:2	TR1BDU:IST:2	

Table 7.2: The 450 channels selected

AHB:IST:2	BHE4TRU:IST:2	CRPH12:IST:2	EICX:IST:2	EICW:IST:2	MHS13:IST1:2	SHA9X:IST:2	CR3IN:IST:2	MHB8O:ILOG:2	MHS13:IST1:2
AHC:IST:2	BHE4TW:IST:2	CRPH145:IST:2	FMD:IST:2	MHS14:IST1:2	MRF500:IST:2	SHB10X:IST:2	CR3OT:IST:2	MHB8R:ILOG:2	
AHD1:IST:2	BHEKPTO:IST:2	CRPH34:IST:2	FMQ:IST:2	MHS15:IST1:2	MRF540:IST:2	SHB7X:IST:2	CR3PN:IST:2	MHB8U:ILOG:2	
AHD2:IST:2	BHEKPTU:IST:2	CRPH41:IST:2	FMW:IST:2	MHS16:IST1:2	MRF580:IST:2	SHD13Y:IST:2	CR3T:IST:2	MHB9:ILOG:2	
AHIMP:IST:2	BHEKTE:IST:2	CRPHFT:IST:2	FMX:IST:2	MHS17:IST1:2	MRF75:IST:2	SHD14X:IST:2	CR3V:IST:2	MHB9L:ILOG:2	
AHPOS:IST:2	CR1IN:IST:2	DHEKTL:IST:2	FMY:IST:2	MHS18:IST1:2	MRF85:IST:2	SHD15Y:IST:2	CR3VBA:IST:2	MHB9O:ILOG:2	
AHSW41:IST:2	CR1OT:IST:2	DHEKTR:IST:2	INKOI:ILOG:2	MHS19:IST1:2	MRFAUS:IST:2	SHD16X:IST:2	CR3VBP:IST:2	MHB9R:ILOG:2	
AHU:IST:2	CR1PN:IST:2	DHEKTVO:IST:2	INKOX:ILOG:2	MHS2:IST1:2	MRTC1WR:IST:2	SHD5Y:IST:2	CR3VRA:IST:2	MHB9U:ILOG:2	
AHV:IST:2	CR1T:IST:2	DHEKTVU:IST:2	KHE0TL:IST:2	MHS20:IST1:2	MRTC1WV:IST:2	SHD6X:IST:2	CR3VRP:IST:2	MHC1:IST:2	
AHWIN:IST:2	CR1V:IST:2	DHELTE:IST:2	KHE0TR:IST:2	MHS21:IST1:2	MRTC2WR:IST:2	SHG21X:IST:2	CR4IN:IST:2	MHC2:IST:2	
BHE1LTA:IST:2	CR1VBA:IST:2	DHELTL:IST:2	KHE1TL:IST:2	MHS22:IST1:2	MRTC2WV:IST:2	TR17BO:IST:2	CR4OT:IST:2	MHC2B:IST:2	
BHE1LTE:IST:2	CR1VBP:IST:2	DHELTR:IST:2	KHE1TO:IST:2	MHS23:IST1:2	MRTC4WR:IST:2	TR17BU:IST:2	CR4PN:IST:2	MHC3:IST:2	
BHE1TOL:IST:2	CR1VRA:IST:2	DHELTW1:IST:2	KHE1TR:IST:2	MHS24:IST1:2	MRTC4WV:IST:2	TR1ACU:IST:2	CR4T:IST:2	MHC4:IST:2	
BHE1TOR:IST:2	CR1VRP:IST:2	DHELTW2:IST:2	KHE1TU:IST:2	MHS25:IST1:2	QHA1:IST:2	TR1BDU:IST:2	CR4V:IST:2	MHC5:IST:2	
BHE1TUL:IST:2	CR2IN:IST:2	EECF1A:ILOG:2	KHE2TELO:IST:2	MHS26:IST1:2	QHA10:IST:2	TR2ACO:IST:2	CR4VBA:IST:2	MHH1LI:ILOG:2	
BHE1TUR:IST:2	CR2OT:IST:2	EECF1B:ILOG:2	KHE2TELU:IST:2	MHS27:IST1:2	QHA3:IST:2	TR2ACU:IST:2	CR4VBP:IST:2	MHH1OI:ILOG:2	
BHE1TWA:IST:2	CR2PN:IST:2	EECF1C:ILOG:2	KHE2TLO:IST:2	MHS28:IST1:2	QHA4:IST:2	TR2BDO:IST:2	CR4VRA:IST:2	MHH1RI:ILOG:2	
BHE1TWE:IST:2	CR2T:IST:2	EECF2A:ILOG:2	KHE2TLU:IST:2	MHS29:IST1:2	QHA9:IST:2	TR2BDU:IST:2	CR4VRP:IST:2	MHH1UI:ILOG:2	
BHE2LTA:IST:2	CR2V:IST:2	EECF2B:ILOG:2	KHE2TRO:IST:2	MHS3:IST1:2	QHB7:IST:2	TR3AO:IST:2	CR5DNO:IST:2	MHI1:ILOG:2	
BHE2LTE:IST:2	CR2VBA:IST:2	EECI:IST:2	KHE2TRU:IST:2	MHS30:IST1:2	QHB8:IST:2	TR3AU:IST:2	CR5DNU:IST:2	MHI10:ILOG:2	
BHE2TL:IST:2	CR2VBP:IST:2	EECKT:IST:2	KHE3TLO:IST:2	MHS31:IST1:2	QHC11:IST:2	TR3BU:IST:2	CR5IN:IST:2	MHI11:ILOG:2	
BHE2TO:IST:2	CR2VRA:IST:2	EECX:IST:2	KHE3TLU:IST:2	MHS32:IST1:2	QHC12:IST:2	TR3D:IST:2</pre>	CR5OT:IST:2	MHI11X:ILOG:2	
BHE2TR:IST:2	CR2VRP:IST:2	EHTI:IST:2	KHE3TRO:IST:2	MHS4:IST1:2	QHG21:IST:2	BHE4TRO:IST:2	CR5PN:IST:2	MHI12:ILOG:2	
BHE2TU:IST:2	CR3DMCO:IST:2	EHTI:IST1:2	KHE3TRU:IST:2	MHS5:IST1:2	QHG22:IST:2	CR2V:IST:2	CR5T:IST:2	MHI13X:ILOG:2	
BHE2TW:IST:2	CR3DMP:IST:2	EHTV:IST:2	KHM1T1:IST:2	MHS6:IST1:2	QHTC13:IST:2	CR2VBA:IST:2	CR5V:IST:2	MHI2:ILOG:2	
BHE3LTA:IST:2	CR3DMPD:IST:2	EHTV:IST1:2	KRE1MT:IST:2	MHS7:IST1:2	QHTC14:IST:2	CR2VBP:IST:2	CR5VBA:IST:2	MHI21:ILOG:2	
BHE3LTE:IST:2	CR3DMW:IST:2	EHTW:IST:2	KRE1OT:IST:2	MHS8:IST1:2	QHTC15:IST:2	CR2VRA:IST:2	CR5VBP:IST:2	MHI21X:ILOG:2	
BHE3TL:IST:2	CR3DSCO:IST:2	EHTW:POSA:2	KRE1UT:IST:2	MHS9:IST1:2	QHTC16:IST:2	CR2VRP:IST:2	CR5VRA:IST:2	MHI22:ILOG:2	
BHE3TO:IST:2	CR3DSDP:IST:2	EHTX:IST1:2	MHB5:ILOG:2	MHTR2E:TRTR:2	QHTC17:IST:2	CR3DMCO:IST:2	CR5VRP:IST:</pre>	MHI23:ILOG:2	
BHE3TR:IST:2	CR3DSDW:IST:2	EHTX:POSA:2	MHB5L:ILOG:2	MHVBON:ALTF:2	QHTC18:IST:2	CR3DMP:IST:2	MHB6RO:ILOG:2	MHI25X:ILOG:2	
BHE3TU:IST:2	CR3DSP:IST:2	EHTX:IST:2	MHB5O:ILOG:2	MRF105:IST:2	QHTC5:IST:2	CR3DMPD:IST:2	MHB6RU:ILOG:2	MHI3RES:ILOG:2	
BHE3TW:IST:2	CR3DSW:IST:2	EICAURA:ILOG:2	MHB5R:ILOG:2	MRF150:IST:2	QHTC6:IST:2	CR3DMW:IST:2	MHB7:ILOG:2	MHI6:ILOG:2	
BHE4LTA:IST:2	CR3IN:IST:2	EICAURB:ILOG:2	MHB5U:ILOG:2	MRF150B:IST:2	SHA11Y:IST:2	CR3DSCO:IST:2	MHB7L:ILOG:2	MHI8:ILOG:2	
BHE4LTE:IST:2	CR3OT:IST:2	EICAURC:ILOG:2	MHB6:ILOG:2	MRF210:IST:2	SHA1Y:IST:2	CR3DSDP:IST:2	MHB7O:ILOG:2	MHI9:ILOG:2	
BHE4TLO:IST:2	CR3PN:IST:2	EICI:IST:2	MHB6LO:ILOG:2	MRF290:IST:2	SHA2X:IST:2	CR3DSDW:IST:2	MHB7U:ILOG:2	MHS1:IST1:2	
BHE4TLU:IST:2</pre>	CR3T:IST:2	EICV:IST:2	MHB6LU:ILOG:2	MRF370:IST:2	SHA3Y:IST:2	CR3DSP:IST:2	MHB8:ILOG:2	MHS10:IST1:2	

Table 7.3: 311 channels without NaN values in the 2019 Data used as base features.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten Semester-, Bachelor- und Master-Arbeit oder anderen Abschlussarbeit (auch der jeweils elektronischen Version).

Die Dozentinnen und Dozenten können auch für andere bei ihnen verfasste schriftliche Arbeiten eine Eigenständigkeitserklärung verlangen.

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

Titel der Arbeit (in Druckschrift):

Performance of different machine learning techniques for forcasting of particle accelerator interlocks

Verfasst von (in Druckschrift):
Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**
Zacharias	Mélissa

Ich bestätige mit meiner Unterschrift:
- Ich habe keine im Merkblatt „Zitier-Knigge" beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

Ort, Datum	**Unterschrift(en)**
Zürich, 13.04.2020	[illegible]

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.