Appendix

1 Data Collection Procedure: OES in low-density plasma with ICP

The collection of spectral data involves several key steps, depending on the strategy followed. For Optical Emission Spectroscopy (OES) in low-density plasma with Inductively Coupled Plasma (ICP), the process typically begins with the generation of the plasma. This is followed by the interaction of the plasma with the sample, which ionizes it (causing particles of the sample to become electrically charged). Due to the de-excitation of these particles, the emitted light can be decomposed and analyzed. This light is collected by a sensor for subsequent spectral analysis. Each of these steps will be explained further in the following section.

Low-Density Plasma Generation: Inductively Coupled Plasma (ICP) In ICP with low plasma density, the goal is to generate a plasma that can ionize the sample of interest for analysis. This is achieved by creating a low-density plasma from an ionized gas, typically argon, through an oscillating electromagnetic field.

Notably, low-density plasma implies a low electron density, typically in the range of 10^9 to 10^{12} electrons/cm^3. Low-density plasma generation is performed at lower power levels, reducing ion collisions and enabling precise processing.

Spectral Decomposition of Light: Using a Grating After ionization, particles of the sample tend to return to their stable configuration, producing light in the process. This light provides information about the elemental composition of the sample.

Following the description proposed by (Fontaine et al. 2023), Figure 5 presents a general framework for the decomposition and capture of light emitted by the ionized sample material at different frequencies. Thus, frequently spectrometer processing the light emitted by a sample is composed of the following components:

1.1 **Collecting Optics:** These components, which can be mirrors or lenses of various shapes, primarily function to focus the incoming light rays onto a focal point. This can be done directly onto a slit or through an optical fiber that guides the light to the slit.

1.2 **Slit:** The slit controls the amount of light that enters the spectrometer by acting as a very small aperture. It reduces the collimated image to a near point, which is then spectrally dispersed. The size of the slit directly affects the resolution of the spectrometer. A smaller slit improves spectral resolution, making it easier to differentiate between closely spaced emission lines and preserving the lineshape. However, this also reduces the amount of light entering the spectrometer, which can be crucial for detecting very dim emissions.

1.3 **Collimating Optics:** These optics ensure that the light rays hit the grating at a uniform angle. This uniformity is essential for the dispersion to depend solely on the wavelength of the light, rather than the angle of incidence.

1.4 **Grating:** The grating is a surface with a very fine, periodic pattern that can operate in either reflection or transmission mode. It functions as an array of thin slits, diffracting the incoming light into its constituent wavelengths.

1.5 **Refocusing Optics:** These optics groups the nearly parallel light rays of the same wavelength onto the sensor, ensuring that each wavelength is accurately captured.

1.6 **Sensor:** The sensor quantifies the number of photons that strike it. Typically, it is a light-sensitive integrated circuit that converts photons into electrons (Charge Coupled Device sensor).

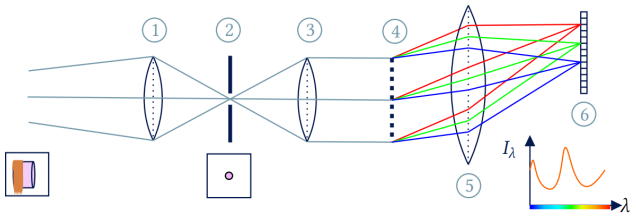


Figure 5: Main components of a generic spectrometer (Fontaine et al. 2023).

Data Collection During data collection, the spectrometer records the intensity of light across a range of wavelengths or frequencies (step 6 of Fig. 5). The emitted light from the sample is captured and presented as a spectrum, with intensity plotted against wavelength or frequency.

Data Processing After data collection, the raw spectral data often requires preprocessing to remove noise, correct for background interference, and enhance signal quality. Common preprocessing steps include baseline correction, smoothing, and normalization. Depending on the method of analysis, the data may also undergo further transformation to extract specific features such as peak intensities, maximum values, or areas under the curve. These features are then used for qualitative or quantitative analysis.

2 Experimental Setup

The experimental setup for data collection within the vacuum chamber is shown in Figure 6 and the schematic overview of all components in Figure 7. The setting used two pumps to achieve vacuum conditions: one allowing to reach 10^{-2} mbar from atmosphere and another up to 10^{-6} mbar.

3 Used Servers

Table 4 lists the specifications for all the servers (with operating system Debian GNU/Linux 12) used in each scenario. For all experiments, standard output was silenced to reduce

potential I/O overhead, and on Servers 1 and 2, the number of threads was limited to 20. All ablation study experiments were carried out exclusively on Server 2. For a complete list of the Python version, libraries, and other specific code requirements, please refer to the README in the GitHub project.

4 Hyperparameter Tuning

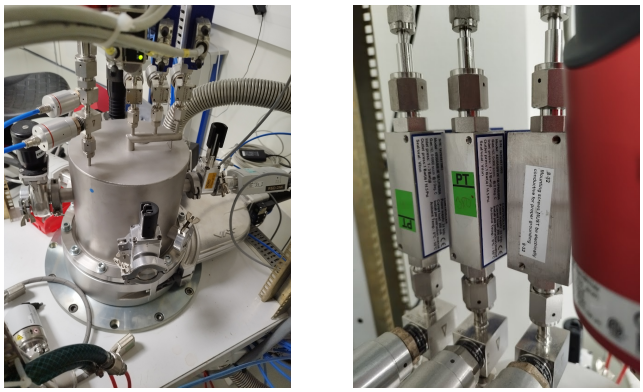
All the default hyperparameter configuration for the tested methods are listed in Table 5.

5 Ablation Study

The mean AUC-PR is compared across three scenarios (A4, A5, and A6) with fixed window sizes of 60, 120, and 240. These results are presented in Tables 6, 7, and 8, respectively.

Figure 8 presents the scores comparison of the best hyperparameter setting of OBKNN and SWKNN by varying the k parameter of the method.

Table 9 presents the overall comparative performance of OBKNN and SWKNN across all tested datasets. It shows the mean AUC-PR performance, along with the standard deviation, over 5 runs for each scenario. Optimal results are presented for OBKNN and SWKNN with window sizes of 60, 120, and 240, as well as various values for the parameter k for SWKNN. The best and second-best scores are highlighted in bold and underlined, respectively.



(a) Vacuum Chamber (b) Leak Valves

Figure 6: Experimental setup of the vacuum chamber for data collection: (left) Setup of the vacuum chamber (right) Leak valves.

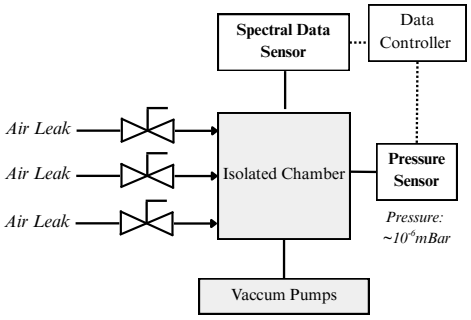


Figure 7: Schema for Spectral Data Collection.

6 Anomaly Score per Method

Figure 9 shows the average anomaly scores from the five runs for each of the tested methods.

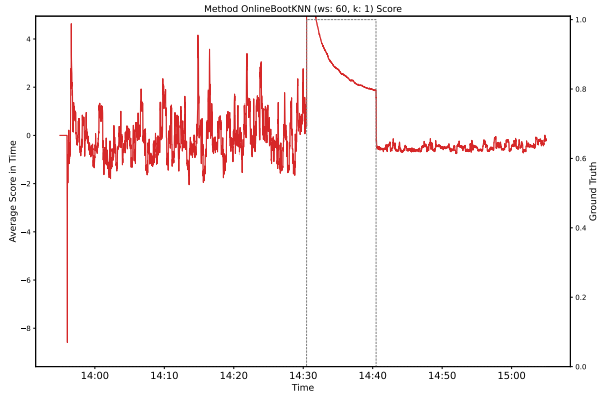
Table 4: Specifications of the computing infrastructure used for experiments.

#	Specifications (CPU)	RAM	Datasets	Restrictions

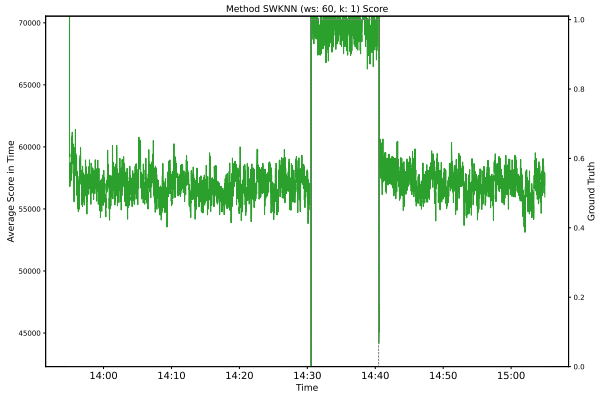
1	16 C / 32 T Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz	512 GB	A1, A2, A3	`os.environ["OMP_NUM_THREADS"] = "20"`
				`os.environ["OPENBLAS_NUM_THREADS"] = "20"`
				`os.environ["MKL_NUM_THREADS"] = "20"`
				`os.environ["NUMEXPR_NUM_THREADS"] = "20"`
				`sys.stdout = open(os.devnull, "w")`

2	16 C / 32 T Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz	512 GB	A4, A5	`os.environ["OMP_NUM_THREADS"] = "20"`
				`os.environ["OPENBLAS_NUM_THREADS"] = "20"`
				`os.environ["MKL_NUM_THREADS"] = "20"`
				`os.environ["NUMEXPR_NUM_THREADS"] = "20"`
				`sys.stdout = open(os.devnull, "w")`

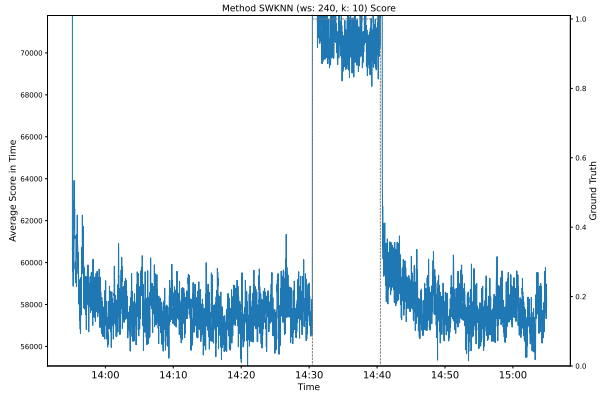
3	24 C / 48 T Intel(R) Xeon(R) Gold 6136 CPU @ 3.00GHz	768 GB	A7, A8, A9	`sys.setrecursionlimit(2000)` `sys.stdout = open(os.devnull, "w")`
4	32 C / 64 T AMD EPYC 9174F@4.1GHz	128 GB	A6	`sys.stdout = open(os.devnull, "w")`



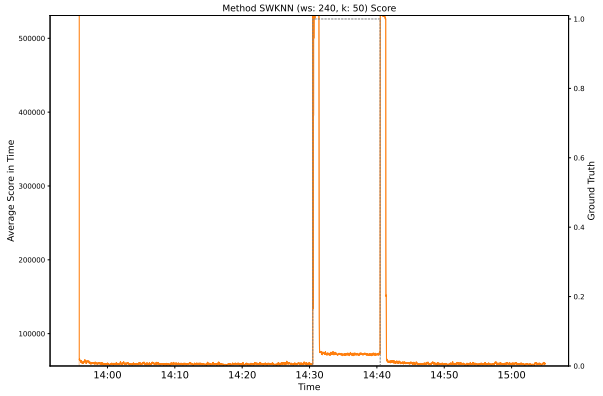
(a) OBKNN ($ws = 60, k = 1$)



(b) SWKNN ($ws = 60, k = 1$)



(c) SWKNN ($ws = 240, k = 10$)



(d) SWKNN ($ws = 240, k = 50$)

Figure 8: Average Anomaly Scores for A6 scenario: OBKNN and SWKNN.

Table 5: Default Hyperparameter Setting with optimized window size for Evaluated Models.

Model	**Parameter**	**Value**
HStree	window_size	60
	anomaly_threshold	0.5
	max_depth	15
	number_of_trees	25
	size_limit	0.1
XStream	window_size	60
	depth	25
	n_chains	100
	num_components	100
RSHash	window_size	120
	decay	0.015
	feature_maxes	[10000]
	feature_mins	[0]
	num_components	100
	num_hash_fns	1
ExactStorm (EStorm)	window_size	120
	max_radius	0.1
OnlineBootKNN (OBKNN)	window_size	120
	algorithm	'brute'
	alpha	0.05
	chunk_size	240
	dmetric	'cityblock'
	ensemble_size	240
	n_jobs	-1
	transf	'ZNORM'
RobustRandomCutForest (RRCF)	window_size	240
	num_trees	4
	tree_size	256
OIF	window_size	240
	growth_criterion	'adaptive'
	max_leaf_samples	32
	n_jobs	-1
	num_trees	32
IForestASD (IFASD)	window_size	240
	initial_window_X	None
KitNet	window_size	240
	hidden_ratio	0.75
	learning_rate	0.1
	max_size_ae	10

Table 6: Mean AUC-PR performance over 3 runs (WS 60).

Method	Avg	Std. Between Scenarios	Std. Within Scenarios
OBKNN	0.933	8.9e-03	2.7e-04
SWKNN ($k = 1$)	0.918	6.4e-02	0.0e+00
SWKNN ($k = 10$)	0.706	2.1e-01	0.0e+00
SWKNN ($k = 50$)	0.477	2.3e-01	0.0e+00

Table 7: Mean AUC-PR performance over 3 runs (WS 120).

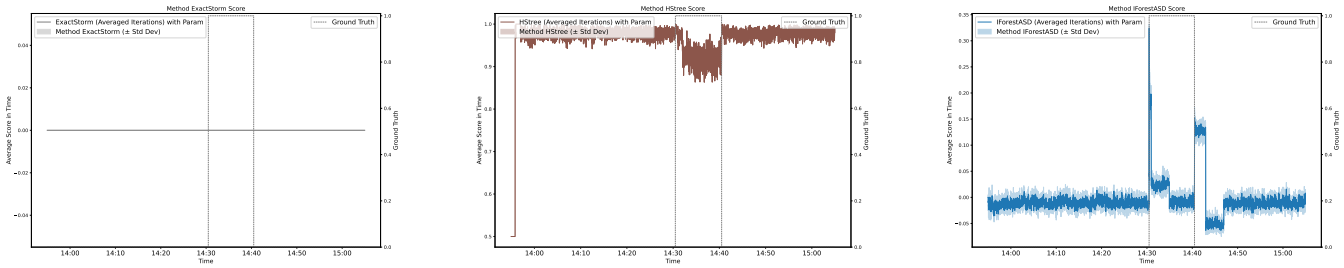
Method	Avg	Std. Between Scenarios	Std. Within Scenarios
OBKNN	0.929	3.1e-03	5.4e-05
SWKNN ($k = 1$)	0.918	6.4e-02	0.0e+00
SWKNN ($k = 10$)	0.745	1.4e-01	0.0e+00
SWKNN ($k = 50$)	0.516	1.7e-01	0.0e+00

Table 8: Mean AUC-PR performance over 3 runs (WS 240).

Method	Avg	Std. Between Scenarios	Std. Within Scenarios
SWKNN ($k = 1$)	0.918	6.4e-02	0.0e+00
OBKNN	0.902	3.6e-02	1.3e-04
SWKNN ($k = 10$)	0.746	1.4e-01	0.0e+00
SWKNN ($k = 50$)	0.517	1.7e-01	0.0e+00

Table 9: Overall Mean AUC-PR ($\pm$ standard deviation) performance OBKNN vs SWKNN over 5 runs for each scenario. The best and second-best scores are highlighted in bold and underlined, respectively.

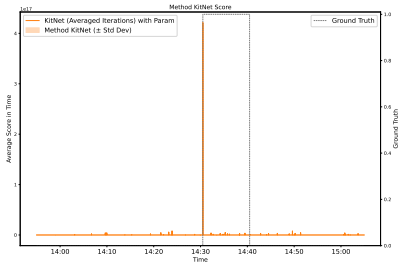
Method	A1	A2	A3	A4	A5	A6	A7	A8	A9	Avg
OBKNN ($k = 1, ws = 120$)	**0.981** $\pm$ 4.4e-06	**0.963** $\pm$ 1.1e-04	<u>0.888</u> $\pm$ 1.9e-04	**0.930** $\pm$ 0.0e+00	<u>0.926</u> $\pm$ 4.4e-05	<u>0.931</u> $\pm$ 5.8e-04	**0.972** $\pm$ 0.0e+00	<u>0.895</u> $\pm$ 2.5e-04	<u>0.928</u> $\pm$ 5.0e-05	**0.935** $\pm$ 1.4e-04
SWKNN ($k = 1, ws = 240$)	<u>0.835</u> -	<u>0.956</u> -	**0.975** -	<u>0.844</u> -	**0.950** -	**0.960** -	<u>0.895</u> -	**0.946** -	**0.965** -	<u>0.925</u> -
SWKNN ($k = 10, ws = 240$)	0.636 -	0.812 -	0.883 -	0.590 -	0.784 -	0.864 -	0.663 -	0.791 -	0.868 -	0.766 -
SWKNN ($k = 50, ws = 240$)	0.334 -	0.554 -	0.683 -	0.338 -	0.540 -	0.673 -	0.343 -	0.545 -	0.675 -	0.521 -



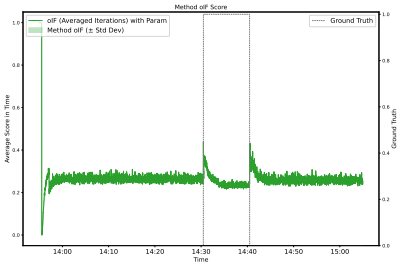
(a) EStorm

(b) HStree

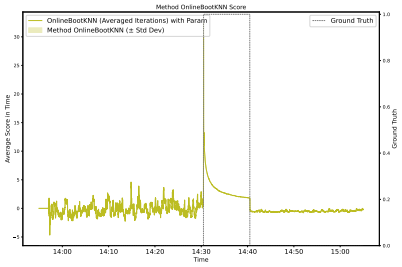
(c) IFASD



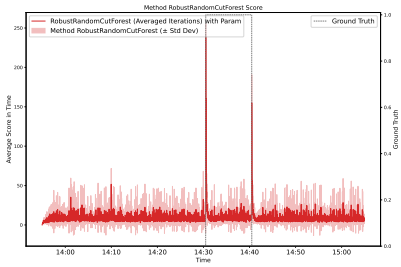
(d) KitNet



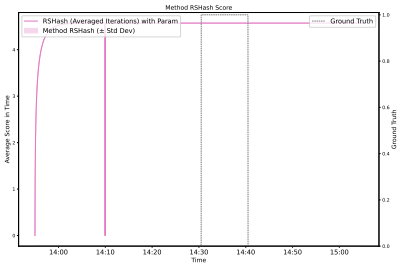
(e) OIF



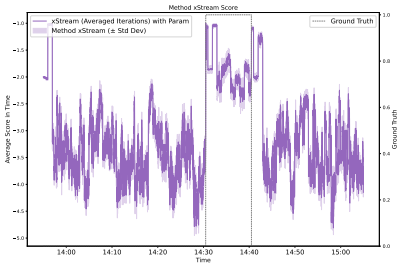
(f) OBKNN



(g) RRCF



(h) RSHash



(i) XStream

Figure 9: Average Anomaly Scores for A6 scenario for all tested methods.