2025 IMEKO TC-6 International Conference on
Metrology and Digital Transformation - **M4DConf 2025**
Benevento, Italy, September 3-5, 2025

"SASO Proficiency Test Machine" - Advanced Pythonic AI Algorithms for Automating and Validating ISO 13528 & ISO 5725-2 at Saudi Standards, Metrology, and Quality Organization-SASO-KSA

Fahad A. AlMuhlaki[1], Saad A. Bin Qoud [1], Rayan A. AlYousefi [1], I. AlFaleh[1], N. Qahtani[1], Khaled AlEnizi[1], AbdulRahman AlMrhom[1], A. El-Matarawey[2]

1. NMCC-SASO- Saudi Arabia
2. National Institute of Standards (NIS-Egypt)

***Abstract* – In the field of laboratory proficiency testing and method validation, linking to international standards such as ISO 13528, ISO 17043 and ISO 5725-2 is essential for ensuring data accuracy, consistency, and inter-laboratory comparability. However, simple implementation these standards can be led to insufficient error-handling and difficult to scale across large datasets. This study introduces a novel Python-based AI framework automating ISO 13528 and ISO 5725-2 compliance, enhancing data integrity and decision-making confidence. We introduce a robust software framework built on modern Python libraries and machine learning techniques that streamline key statistical computations required by these standards—including outlier detection, repeatability and reproducibility assessment, and performance scoring. The system not only automates routine calculations but also introduces intelligent validation checks that flag anomalies, inconsistencies, or deviations from expected patterns, thereby enhancing data integrity and decision-making confidence. By integrating AI-enhanced analytics with standardized evaluation protocols, this work bridges the gap between traditional statistical methods and modern computational capabilities.**

***Keywords*:** Python, AI algorithms, ISO 13528, ISO 5725-2, automation, statistical validation, proficiency testing, data quality, SASO, NMCC

I. INTRODUCTION

Proficiency testing (PT) and method validation are critical components of laboratory quality assurance programs, ensuring that measurement results are accurate, reliable, and comparable across different facilities.

The objectives of PT are:

- Assess the performance of the participating laboratories in the scope of the PT scheme.
- Provide technical support to the participating laboratories.
- Identify the technical issues in the performance of the participating laboratories and recommend potential solutions.
- Provision of additional confidence in services.
- Assess and validate the uncertainty claimed by the participants.
- Identify the sources of error in their performance.
- Support the participating laboratories with technical/statistical recommendation to improve their performance.

Standards such as ISO 13528 [1], ISO 5725-2[2] and ISO 17043[3] provide detailed guidelines for evaluating laboratory performance and validating measurement methods. However, traditional approaches to implementing these standards often rely on manual calculations or spreadsheet-based tools, which can be labor-intensive, error-prone, and difficult to scale. The advent of modern programming languages like Python offers an opportunity to streamline and automate the application of these standards. By applying robust statistical techniques, visualizations and efficient computational tools, laboratories can achieve faster, more consistent results while minimizing human error. Automation helps ensure consistent interpretation of laboratory performance and measurement accuracy across multiple datasets. This paper explores the development of a Python-based AI automation framework for implementing key aspects of ISO 13528 and ISO 5725-2, with a focus on robust statistical methods such as median-based estimators and outlier-resistant calculations. Through this approach, we aim to demonstrate how automation enhances the efficiency, accuracy, and transparency of proficiency testing and method

validation processes. In many real-world scenarios, laboratory data may contain outliers due to equipment malfunctions, human error, or environmental variability. Classical statistical measures like the mean and standard deviation are highly sensitive to such outliers. To address this, ISO 13528 recommends the use of robust estimators. These provide more reliable central tendency and dispersion estimates, ensuring that performance evaluations remain accurate even when data is not perfectly normal. Python [4], with its rich ecosystem of scientific computing packages, provides an ideal platform for implementing the statistical requirements of ISO 13528 and ISO 5725-2. Libraries such as NumPy [5] and SciPy [6] offer built-in functions for calculating z-scores, zeta score. Normalized En value, repeatability limits, and outlier detection criteria. Pandas [7] simplifies data manipulation, while Matplotlib [8] and Seaborn [9] allow for clear visual reporting of results. Together, these tools enable the development of fully automated pipelines for evaluating laboratory performance and validating measurement methods.

II. MECHANISM OF STATISTICAL MODEL

Figure 1. illustrates a structured process flow for the development, testing, and evaluation of calibration/test materials, as outlined by relevant ISO [1,3] standards. This flowchart provides a comprehensive overview of the steps involved in ensuring the quality and reliability of such materials, which are critical for accurate measurements and laboratory performance assessments. All figures that demonstrated in this work are coming from different examples.

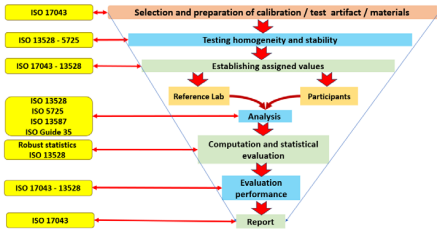

Fig. 1. Statistical Model Flowchart.

II.I. SELECTION / PREPARATION OF CALIBRATION/TEST MATERIALS

The process begins with the selection and preparation of calibration or test materials according to [3]. This step ensures that the equipment, device, artifact or materials chosen are suitable for their intended purpose and meet the necessary specifications. Proper preparation is crucial to maintain consistency and accuracy throughout subsequent stages.

II.II. TESTING HOMOGENEITY AND STABILITY

Once the materials are prepared, they undergo rigorous testing according to [1, 2] to evaluate their homogeneity (uniformity across samples) and stability (consistency over time). These tests are essential to confirm that the materials will provide reliable results during use.

We validate the homogeneity using both models of ISO 13528, ISO 5725-2 and analysis of variances (ANOVA).

II.II.A. OUTLIER

The algorithm checks the outlier by series of tests includes Cochran and Grubbs tests as shown in figure 2.

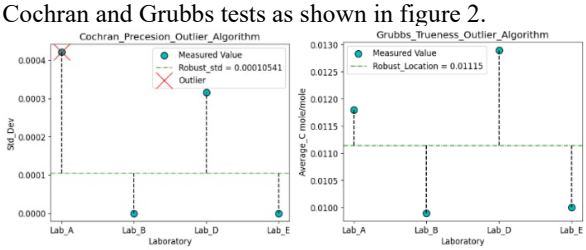

Fig. 2. Outlier detection algorithm

II.II.B. Homogeneity

The algorithm checks the homogeneity by investigate the between and within laboratories standard deviations, figure 3. The criterion of homogeneity is $s_s \leq 0.3\sigma_{pt}$ by compare the between-sample standard deviation s_s with the standard deviation for proficiency assessment σ_{pt}. The criteria $s_s \leq 0.3\sigma_{pt}$ or $s_s \leq 0.1\delta_E$ are used to determine whether the proficiency test items are adequately homogeneous, δ_E represents the target value difference or the minimum detectable difference between samples that is considered significant for the proficiency test. It is often based on the performance requirements or the intended use of the test. These criteria ensure that the variability among samples does not significantly affect the performance evaluation.

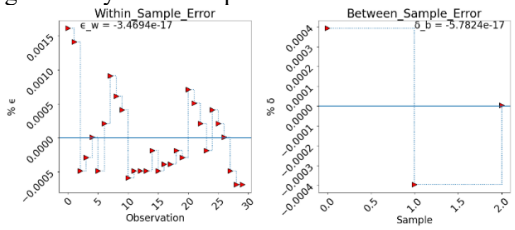

Fig. 3. Between-Within labs standard deviation algorism.

If the homogeneity criterion is satisfied, algorithm used directly σ_{pt} but if not, an extension parameter has been added to compensate if z-score was selected.

II.II.C. STABILITY

The algorithm checks the stability as:

a. If the protocol is used z score (test, samples, chemicals, concentration, etc..), algorithm investigate the difference between averages based on the distributions as $|\overline{y_1} - \overline{y_2}| \leq 0.3\ \sigma_{pt}$ and its expanding form $|\overline{y_1} - \overline{y_2}| \leq 0.3\ \sigma_{pt} + 2\sqrt{u_{\overline{y_1}}^2 - u_{\overline{y_2}}^2}$.

b. If the protocol used En or zetta score (calibration, equipment, device, etc..), algorithm investigate the stability by three parallel techniques.

- *Allowance error*: A suitable statistical criterion was established in order to ensure that the changes in the artifact's results are within the uncertainty and have no

impact on the evaluation of performance for the participants. $\delta_s = |Y_2 - Y_1| \leq U_{Ref}$

- *Trend analysis*: linear regressing fitting, standard error of fit is used as a compensate factor for reference lab uncertainty, figure 4.

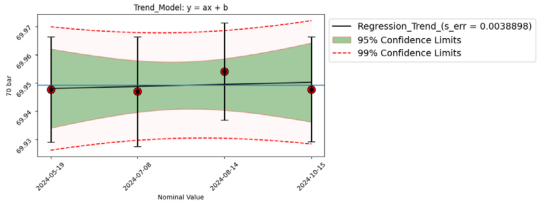

Fig. 4. Trend model.

- *Cumulative sum control chart (CUSUM QC Chart)*. The Algorithm sets the limits to $\pm 2\sigma$ to detect the cumulative error that exceeds the 95% confidence level. The QC was applied to the results reported by the reference laboratory at each nominal point, figure 5.

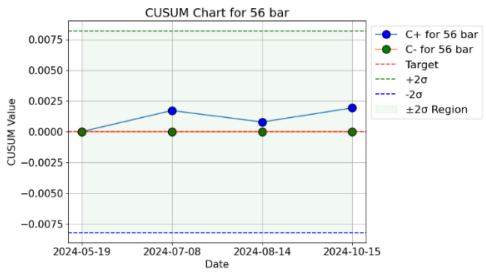

Fig. 5. CUSUM QC Chart.

II.III. ESTABLISHING ASSIGNED VALUES

After confirming homogeneity and stability, assigned values are established for the materials. This involves determining the true or reference values that the materials should exhibit. This step is critical for ensuring that the materials can serve as reliable references in calibration and testing processes. The algorithm treated this part as "if statement coding".

a. If assigned values are coming from reference lab, kernel density plot is used to measure the tendency to center location, it provides a smooth curve describing the general shape of the distribution of a data set (figure 6). The idea underlying the kernel estimate is that each data point is replaced by a specified distribution (typically normal), centered on the point and with a standard deviation bandwidth σ_k. In general and examination inspection, we used $\sigma_k = 0.9\frac{s^*}{p^{0.2}}$, $\sigma_k = 0.25\delta_\varepsilon$ respectively. uncertainty associated with the reference values was evaluated considering the uncertainty of the reference laboratory and the precision of the measurements. The combined standard uncertainty was evaluated as $u_c = \sqrt{u_s^2 + u_{NMCC}^2}$

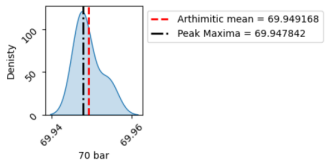

Fig. 6. Kernel density plot.

b. If assigned values are coming from robust statistics.

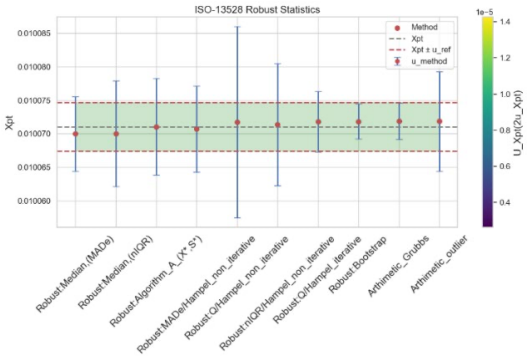

Fig. 7. ISO-13528 robust statistics techniques.

II.IV. REFERENCE LABORATORY AND PARTICIPANT ANALYSIS

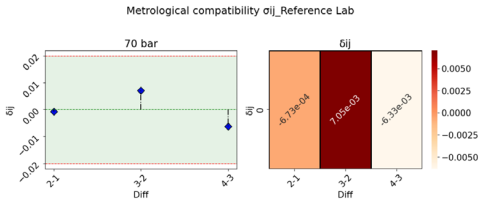

Fig. 8. Metrological compatibility criterion.

Participants: Other laboratories or participants also analyze the same materials independently. Their results are compared against those from the reference laboratory to assess consistency and identify any discrepancies.

II.IV.A. TRUENESS ASSESSMENT

In this section, the error in the measurements results reported by the participants (i.e., the deviation from the reference value) was assessed regardless of the uncertainty reported by the participants. The aim of this section is to investigate whether the measurements conducted by the participants require further correction, figure 9. The error in the results reported by the participant was compared with the reference value. As the results reported by the participants are within the tolerance limits, it indicates a good trueness for the results reported by the participant.

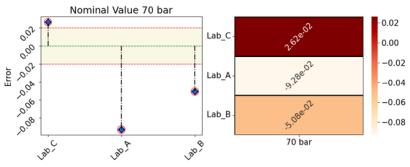

Fig. 8. Trueness Assessment.

II.IV.B. ACCURACY ASSESSMENT

In this section, the deviations of the results reported by the participants are assessed considering the uncertainty reported by the participants. The results reported by the participants were graphically represented, while the expanded uncertainties of the reference values were represented as tolerance lines. Figure 9 illustrates the accuracy assessment for the results reported by the participants. The uncertainty line should be intersected with red line otherwise the accuracy of this measurement is considered to be poor. The laboratory shall consider the comments and recommendations given in this and previous sections and set the appropriate correction/corrective actions for these issues.

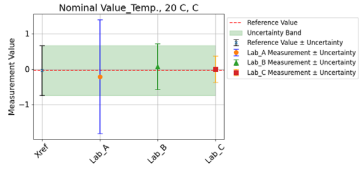

Fig. 9. Accuracy Assessment.

II.IV.C. UNCERTAINTY ASSESSMENT

The uncertainty values reported by the participants have been assessed using the repeatability plot. In this statistical analysis, confidence intervals are established based on the reference value and its associated standard uncertainty. The results of laboratories that lie outside these confidence intervals are considered significantly different from the reference values and corrective/correction actions shall be considered by the corresponding laboratories, figure 10.

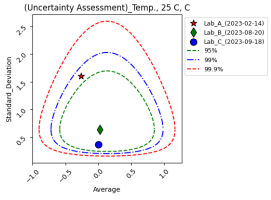

Fig. 10. Uncertainty Assessment.

II.V. COMPUTATION AND STATISTICAL EVALUATION

To check how consistent the errors were across different participants, algorithm used a method developed by Mandel, which is outlined in [2]. This method helps us understand how reliable and repeatable the measurements are when different labs use the same technique. The goal is to see how much the results vary from one lab to another and whether these variations are within an acceptable range. For each lab, we calculate two key numbers: one that shows how precise their measurements are (how close repeated measurements are to each other), and another that shows how accurate they are (how close their measurements are to the true value). This allows us to identify any labs that might have issues with their measurement methods or equipment, figure 11.

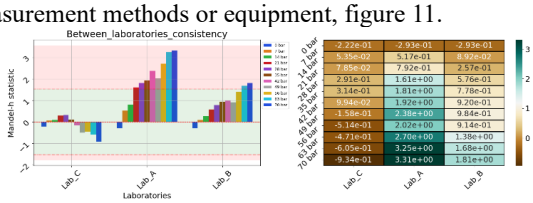

Fig. 11. Mandel's statistic.

II.VI. EVALUATION OF PERFORMANCE

Based on the statistical evaluations of [1, 3], the performance of the materials and the participating laboratories is assessed. This includes evaluating how well the materials meet the required specifications and how accurately the laboratories perform their analyses. Feedback from this evaluation helps identify areas for improvement and ensures that the materials, equipment, device or artifact are fit for purpose, figures 12, 13 for En Score evaluation and Z-score evaluation respectively.

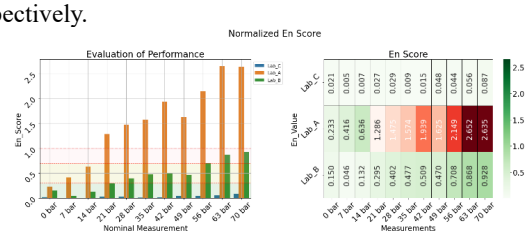

Fig. 12. Normalized En Score Evaluation.

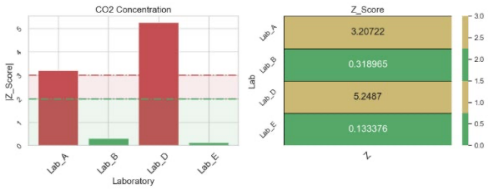

Fig. 13. Z-Score Evaluation.

II.VII. REPORTING

The final step involves generating a comprehensive report summarizing the entire process according to the requirements of ISO 17043. This report documents the results of the homogeneity and stability tests, assigned values, statistical evaluations, and performance assessments. Each step is designed to enhance the reliability and accuracy of the

artifacts, ultimately supporting high-quality measurement practices in various scientific and industrial applications. The systematic approach not only ensures compliance with international standards but also fosters transparency and reproducibility in laboratory operations.

III. PYTHON SCRIPT CODE

Python [4] has gained significant traction as one of the most popular programming languages, largely due to its simplicity, adaptability, and comprehensive ecosystem. The language's syntax is intentionally designed to prioritize clarity and brevity, enabling users to write code that is both intuitive and efficient, which is particularly advantageous for beginners and experts [10]. Python's extensive library support, including tools like NumPy, Pandas, and TensorFlow, offers powerful capabilities for tasks such as data manipulation, artificial intelligence, and automation [11, 12]. Our code consists of more than 1200, 584 and 682 lines of code for calculations, training the model and validation respectively. We introduced here some examples and welcome for any request, help and support to our colleagues or reader of this article around the world.

III.I. STABILITY / HOMOGENEITY CODE

To assess homogeneity, we build a homogeneity check function (*homogeneity_check*) on our dataset, which included multiple samples from two groups (e.g., Sample 1 and Sample 2). The homogeneity test evaluates whether the variance across groups is consistent, using a specified standard deviation (*sigma_pt*) as a threshold for proficiency assessment, figure 14. The results of this test were printed to confirm whether the data met the criteria for homogeneity.

```
# Example data for homogeneity check
homogeneity_data = [
    [10.1, 10.2, 10.3],  # Sample 1 replicates
    [10.4, 10.5, 10.6],  # Sample 2 replicates
    # ... add more samples
]
sigma_pt = 0.5  # Proficiency assessment SD

# Run homogeneity check
hom_result = homogeneity_check(homogeneity_data, sigma_pt)
print(f"Homogeneity Check: s_s={hom_result['s_s']:.4f}, s_w={hom_result['s_w']:.4f}")
print(f"Homogeneous? {'Yes' if hom_result['homogeneous'] else 'No'}")

# Example data for stability check
y1 = 10.2  # Average before
y2 = 10.3  # Average after
u_y1 = 0.1  # Uncertainty in y1
u_y2 = 0.1  # Uncertainty in y2

# Run stability check
stab_result = stability_check(y1, y2, sigma_pt, u_y1, u_y2, use_expanded=True)
print(f"Stability Check: Difference={stab_result['difference']:.4f}, Criterion={stab_result['criterion']:.4f}")
print(f"Stable? {'Yes' if stab_result['stable'] else 'No'}")

# Cochran's test for outlier detection
cochran_result = cochran_test(homogeneity_data)
print(f"Cochran's Test: Stat={cochran_result['cochran_stat']:.4f}, Critical={cochran_result['critical_value']:.4f}")
print(f"Outlier detected? {'Yes' if cochran_result['outlier_detected'] else 'No'}")
```

```
Homogeneity Check: s_s=0.2041, s_w=0.1000
Homogeneous? No
Stability Check: Difference=0.1000, Criterion=0.4328
Stable? Yes
Cochran's Test: Stat=0.5000, Critical=7.7086
Outlier detected? No
```

Fig. 14 Stability / homogeneity code.

Next, we conducted a stability check to evaluate the consistency of measurements over time or under varying conditions. This involved comparing two sets of measurements (y_1 and y_2) with their respective uncertainties (u_y_1 and u_y_2). The stability check function (*stability_check*) calculated the difference between the means of the two datasets and determined whether the observed variation was within acceptable limits. The output indicated whether the data was deemed "Stable" based on predefined criteria. Finally, to identify any potential outliers in our dataset, we applied Cochran's test for outlier detection (*cochran_test*). This test evaluates whether any single observation significantly deviates from the rest of the dataset. The critical value obtained from the test was compared against a pre-defined threshold to determine whether an outlier was detected. The results of these analyses ensured that our data met the necessary statistical standards for further processing and interpretation.

III.II. DECISION TREE TRAINING CODE

The decision tree [13] is a supervised machine learning algorithm [14-19] used for both classification and regression tasks. It works by recursively partitioning the dataset into subsets based on feature values, creating a tree-like structure where each internal node represents a decision rule (e.g., "Is feature X > threshold?"), figure 15.

```
data = data
X = data.data  # Features
y = data.target  # Labels

# Split the dataset into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Decision Tree Classifier
clf = DecisionTreeClassifier(max_depth=3, random_state=42)  # Limiting depth to prevent o

# Train the model on the training data
clf.fit(X_train, y_train)

# Make predictions on the test data
y_pred = clf.predict(X_test)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

Fig. 15 Decision Tree training code.

III.III. Z-SCORE CODE

The code snippet and accompanying plot illustrate the process of analyzing a dataset using statistical methods to assess its distribution and identify key quantiles. The first step involves calculating z-scores for a sample dataset (*sample_props_20*). Z-scores are computed using the formula $z = \frac{lab_i - \mu}{\sigma}$ where μ is the assigned, reference, consensus value based on method used and σ is the standard deviation, figure 16.

```
zscores = [(value - mu) / std for value in sample_props_20]
quantile_z=np.quantile(zscores, [0.025,0.15, 0.5,0.85 , 0.975])

# Plot the histogram.
plt.hist(zscores, bins=100, density=True, alpha=0.6, color='g')
# Plot the PDF.
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax)
p = norm.pdf(x, 0, 1)
plt.plot(x, p, 'r', linewidth=2)
plt.axvline(x=quantile_z[0], color='b', linewidth=1.5, dashes=[6, 2])
plt.axvline(x=quantile_z[1], color='b', linewidth=1.5, dashes=[6, 2])
plt.axvline(x=quantile_z[2], color='b', linewidth=1.5, dashes=[6, 2])
plt.axvline(x=quantile_z[3], color='b', linewidth=1.5, dashes=[6, 2])
plt.axvline(x=quantile_z[4], color='b', linewidth=1.5, dashes=[6, 2])
```

```
<matplotlib.lines.Line2D at 0x1d20e82b490>
```

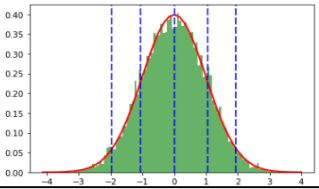

Fig. 16 Z-Score Code.

IV. MODEL VALIDATION

Annex E (Informative) in [1] provides a set of examples designed to validate the procedures outlined in the main document. We have successfully validated all 15 examples and compared the results from our model with those mentioned in the standard. Our results fully match the standard values, with even greater precision in the number of significant digits. For simplicity, we present only two of these validated examples here.

E.3 Comprehensive example of atrazine in drinking water

A proficiency testing scheme for Atrazine in drinking water has 34 participants. All coming section will match and compare the values that are calculated by ISO 13528 [1] and SASO model. According to the table E4 in [1], figures 17 and 18 are described the sorted data increment in standard and in SASO model.

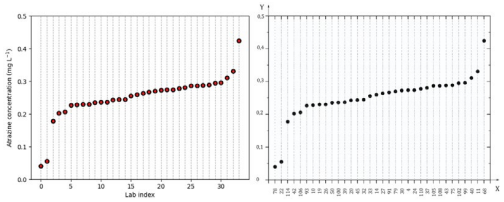

Fig. 17 SASO Model.

Fig. 18 ISO 13528 Model.

The Kernel density plot for participant results is shown in figures 19, 20.

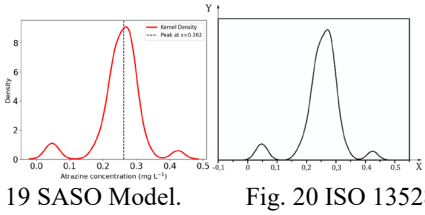

Fig. 19 SASO Model.

Fig. 20 ISO 13528 Model.

ISO 13258 model provide the calculated values of location, standard deviation and uncertainty, SASO model match those numbers with performance 99.98% as in table 1.

Table 1. Summary Statistics of SASO Model

Robust: Median, nIQR	[52]: Median_ 0.262	[57]: nIQR 0.040234057499999	[58]: u_nIQR 0.00862510491
Robust: Median, MADe	[52]: Median_ 0.262	[53]: MADe 0.038557999999999	[58]: u_nIQR 0.00862510491
Robust: Algorithm A (x*, s*)	[70]: itr_alga 0.257016	[69]: itr_alga(df.xi,s 0.039495557070035	[71]: u_alg_A 0.0084832722
Robust: Q/Hampel	[103]: Results.Xpt 0.259633141	[107]: Results.opt.iloc 0.0425749999997	[111]: Results['u_Xpt 0.00919740426
Bootstrap (for mean)	[104]: Results.) 0.2503274	[113]: Results.opt.iloc[4 0.066768844	[115]: Results['u_Xpt 0.011583478251
Arithmetic, outliers removed	[41]: np.mean(Gru 0.258761290	[117]: Results.opt.iloc[5 0.033683012118	[118]: Results['u_Xpt'] 0.006147504708
Arithmetic, outliers included	[105]: Arth_out[6 0.2512117	[120]: Results.opt.iloc[6 0.0672108160699707	[118]: Results['u_Xp 0.0115347

CONCLUSION

A novel smart algorithm has been developed to automate the implementation of key standards, including ISO 13528, ISO 5725-2, and ISO 17034, leveraging machine learning techniques implemented in Python. This algorithm demonstrates exceptional performance by successfully validating all 15 examples provided in ISO 13528, achieving nearly complete alignment with the standard's results or, in some cases, even surpassing their accuracy. The robustness and precision of the algorithm underscore its potential as a reliable tool for automating complex statistical procedures in proficiency testing and interlaboratory comparisons. Looking ahead, the next steps involve enhancing its accessibility and scalability by integrating the algorithm into a FastAPI framework and reprogramming the codebase to function as a web-based application.

REFERENCES

1. ISO 13528:2022 Statistical methods for use in proficiency testing by interlaboratory comparison.
2. ISO 5725-2:2019 Accuracy (trueness and precision) of measurement methods and results - Part 2: Basic method for the determination of repeatability and reproducibility of a standard measurement method.
3. ISO/IEC 17043:2023 Conformity assessment - General requirements for the competence of proficiency testing providers.
4. Python Software Foundation, *Python.* Available: https://www.python.org.
5. NumPy Developers, *NumPy.* Available: https://numpy.org.
6. SciPy Developers, *SciPy.* Available: https://scipy.org.
7. Pandas Development Team, *Pandas.* Available: https://pandas.pydata.org.
8. Matplotlib Development Team, *Matplotlib.* Available: https://matplotlib.org.
9. Seaborn Development Team, *Seaborn.* Available: https://seaborn.pydata.org.
10. Python Software Foundation, *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace Independent Publishing Platform, 2009. ISBN: 978-1-4414-1269-0.
11. T. E. Oliphant, "Python for Scientific Computing," *Computing in Science & Engineering,* vol. 9, no. 3, pp. 10-20, 2007. DOI: 10.1109/MCSE.2007.58.
12. W. McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and I Python. Sebastopol, CA: O'Reilly Media, 2010.
13. Scikit-learn Developers, *Decision Trees in Scikit-learn.* Available: https://scikit-learn.org/stable/modules/tree.html.
14. J. R. Quinlan, "Induction of Decision Trees," *Machine Learning,* vol. 1, no. 1, pp. 81-106, 1986, doi: 10.1007/BF00116251.
15. T. M. Mitchell, *Machine Learning* . New York, NY, USA: McGraw-Hill, 1997.
16. C. M. Bishop, *Pattern Recognition and Machine Learning* . New York, NY, USA: Springer, 2006.
17. S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach,* 3rd ed. Upper Saddle River, NJ, USA: Pearson Education, 2010.
18. L. Breiman, "Statistical Modeling: The Two Cultures," *Statistical Science* , vol. 16, no. 3, pp. 199-231, 2001, doi: 10.1214/ss/1009213726.
19. V. N. Vapnik, *The Nature of Statistical Learning Theory.*