**E**

What is the difference between standard deviation and standard error or the mean ? Can you explain it in a simple yet intuitive way ?

The standard deviation and standard error of the mean are related but different statistical measures. Let me explain them using a simple example:

Imagine you want to know the average height of students in a class. You measure the heights of all 30 students and calculate the mean (average) height, let's say it's 170 cm.

The standard deviation tells you how spread out the individual heights are from that mean value of 170 cm. If the standard deviation is small, say 3 cm, it means that most students have a height close to 170 cm. But if the standard deviation is large, say 10 cm, it means the heights are more spread out - some students are quite shorter and some quite taller than the mean.

So the standard deviation measures the typical amount that an individual data point deviates from the mean. A small standard deviation indicates the data points are clustered close to the mean, while a large standard deviation indicates the data is more spread out.

The standard error of the mean, on the other hand, tells you how precise the mean (170 cm) is as an estimate of the true population mean height. It takes into account both the standard deviation and the sample size (number of students measured).

With a larger sample size, say 100 students instead of 30, the standard error shrinks because the mean becomes a more precise estimate of the true population mean height. But regardless of sample size, a smaller standard deviation will also yield a smaller standard error.

In essence:

* Standard deviation measures spread of individual data points around the mean
* Standard error measures uncertainty around the mean itself

A small standard deviation and large sample size will give you a small standard error, meaning your mean is a precise estimate of the true population parameter.

Copy

**E**

What is the population mean in your example?

In my example, I did not explicitly specify the population mean. The population mean refers to the true average height for the entire group or population that the sample is taken from.

In the example:

* The sample was the 30 (or 100) students whose heights were measured
* The sample mean was calculated as 170 cm based on those measured heights

However, this sample mean of 170 cm is only an estimate of the true population mean height for all students (say in a school, city or country).

The population mean is the parameter representing the actual average height if you could measure the heights of every single individual in the entire population.

Since realistically we cannot measure the entire population, we take a sample and use the sample mean as an estimate of the unknown population mean parameter.

The standard error of the mean tells us how precise or accurate this sample mean of 170 cm is likely to be in estimating the true, unknown population mean height parameter.

A smaller standard error suggests the sample mean of 170 cm is likely very close to the true population mean height.

Copy

**E**

Thank you! When I implement nested cross validation to evaluate a machine learning algorithm for a classification problem what is best to compute over the multiple rounds of the outer loop (train-test splits)? The Std or the SEM?

When implementing nested cross-validation to evaluate a machine learning algorithm for classification, it is generally recommended to compute the standard error of the mean (SEM) over the multiple rounds of the outer loop splits.

The key reasons are:

1. The outer loop splits are meant to estimate the generalization performance of the model on unseen test data. Each outer loop provides an approximately unbiased estimate of the true test performance.
2. By computing the SEM across these multiple outer loop estimates, you can get a measure of the uncertainty or precision in estimating the true test performance metric (e.g. accuracy, F1-score etc.).
3. The standard deviation (std) alone does not account for the number of outer loop splits. With the same std, having more outer splits will reduce the SEM and therefore increase the precision of the estimated metric.
4. Reporting the mean performance metric along with the SEM provides important information about the expected variability of the model performance when evaluated on new test data.
5. Most well-established evaluation papers and benchmarks report the mean and SEM when using cross-validation for model evaluation.

That said, computing and reporting both the std and SEM can also be informative, with the std giving a sense of the spread in individual test performance across splits.

In summary, by computing the SEM over the outer loop splits of nested CV, you can robustly estimate and report the mean model performance along with the uncertainty interval, which is highly useful for model evaluation and comparison.

Copy

**E**

Let's say that we perform N repetitions (rounds ) of the outer loop train-test split and therefore get N values for a performance metric m. What would be the formula for estimating std and SEM in this case for that metric?

Sure, let's go through the formulas for computing the standard deviation (std) and standard error of the mean (SEM) when you have N repetitions of the outer loop train-test split, resulting in N values for a performance metric m.

Let the N values be: m1, m2, m3, ..., mN

1. Standard Deviation (std): The standard deviation measures the spread of the data points around the mean value. The formula is:

std = sqrt( Σ(mi - mean(m))^2 / (N-1) )

Where:

* mi is the individual performance value
* mean(m) is the arithmetic mean of the N values
* Σ represents summation over i=1 to N
* N-1 is used as the denominator for an unbiased estimate

1. Standard Error of the Mean (SEM): The SEM measures the uncertainty in estimating the true population mean from a sample mean. The formula is:

SEM = std / sqrt(N)

Alternatively, you can calculate SEM directly as:

SEM = sqrt( Σ(mi - mean(m))^2 / (N\*(N-1)) )

So in summary:

1. First calculate the mean(m) as the arithmetic average of the N values.
2. Then calculate the std using the squared deviation from mean(m) formula above.
3. The SEM is then calculated as std/sqrt(N)

The SEM will always be smaller than or equal to the std. As N increases, the SEM becomes smaller, reflecting greater precision in estimating the true mean performance.

When reporting, it is typical to state: "Mean performance = X +/- Y" Where X is mean(m) and Y is the SEM

This communicates the estimated mean performance along with the uncertainty interval around that mean estimate.

Copy

**E**

What if we use repeated nested cross validation in which we repeat the nCV outer loop (having N rounds) M times, each time with a different seed? How that would change the formulas above?

If we use repeated nested cross-validation, where the entire nested CV procedure (with N rounds of the outer loop) is repeated M times with different seeds, the formulas for calculating the standard deviation (std) and standard error of the mean (SEM) need to be adjusted.

In this case, we have M sets of N performance values, resulting in a total of M\*N data points.

Let's denote the M sets of N performance values as:

Set 1: m11, m12, m13, ..., m1N Set 2: m21, m22, m23, ..., m2N ... Set M: mM1, mM2, mM3, ..., mMN

To calculate the overall std and SEM across all M\*N data points:

1. Standard Deviation (std):

std = sqrt( Σ(mij - mean(m))^2 / ((M\*N)-1) )

Where:

* mij is the performance value in set i and round j
* mean(m) is overall mean across all M\*N values
* Σ represents summation over i=1 to M and j=1 to N

1. Standard Error of the Mean (SEM):

SEM = std / sqrt(M\*N)

Alternatively, you can calculate SEM directly as:

SEM = sqrt( Σ(mij - mean(m))^2 / ((M*N)*(M\*N-1)) )

So essentially, you are treating all the M\*N performance values as a single pooled sample to calculate the overall std and SEM.

The key difference from the single nested CV case is that the SEM is now divided by sqrt(M\*N), which accounts for the total number of data points across all repeated runs.

As M and N increase, the SEM will decrease, indicating higher precision in estimating the true mean performance metric due to the repeated evaluations.

This allows you to robustly quantify and report the overall mean performance along with its uncertainty by stating: "Mean = X +/- Y" where X is mean(m) and Y is the SEM calculated over all M\*N values.

Copy

**E**

How is SEM related to confidence intervals ?

The standard error of the mean (SEM) is directly related to constructing confidence intervals around the estimated mean.

A confidence interval provides a range of values that is likely to contain the true population mean with a certain probability.

The formula to calculate a confidence interval is:

Confidence Interval = Mean +/- (Critical Value) \* (SEM)

Where:

* Mean is the sample mean
* SEM is the standard error of the mean
* Critical Value depends on the specified confidence level

Common confidence levels are 95% and 99%.

The critical values relate to how many standard errors away from the mean you need to go to capture the true mean with the specified confidence. For example, the 95% confidence level has a critical value of 1.96.

So a 95% confidence interval would be: Mean +/- 1.96 \* SEM

This interval has a 95% probability of containing the true population mean.

A smaller SEM will result in a narrower confidence interval. As the SEM decreases (e.g. from a larger sample size), the interval becomes more precise in estimating the true mean.

The relationship is that the SEM determines the radius of the confidence interval around the mean estimate. Roughly 95% of the time, the true mean should fall within 1.96 standard errors of the sample mean.

So reporting the SEM allows you to easily calculate explicit confidence intervals around your estimated mean for any desired confidence level, providing a measure of uncertainty.

Copy

**E**

Is all that based on some gaussianity assumption or not?

Yes, the use of standard error of the mean (SEM) to construct confidence intervals around the estimated mean is based on the assumption that the data follows a normal (Gaussian) distribution.

Specifically, it relies on the following key assumptions:

1. Normality - The population from which the sample is drawn follows a normal distribution.
2. Independence - The observations in the sample are independent of each other.
3. Known standard deviation or large sample size - Either the population standard deviation is known, or the sample size is large enough (typically n>30) for the Central Limit Theorem to apply.

When these assumptions hold, then the sampling distribution of the mean will follow a normal distribution. This allows us to use critical values from the standard normal distribution (e.g. 1.96 for 95% confidence) to calculate confidence intervals based on the SEM.

If the normality assumption is violated, especially for small sample sizes, then different methods may be required to obtain accurate confidence intervals, such as:

1. Data transformations to achieve normality
2. Use of non-parametric methods like bootstrapping
3. Relying on other probability distributions like the t-distribution

However, for large sample sizes, the Central Limit Theorem ensures that the sampling distribution of the mean will be approximately normal, even if the population distribution is not. In such cases, confidence intervals based on SEM remain reasonably accurate.

So in summary - using SEM for confidence intervals is valid when the normality assumption holds or when sample sizes are sufficiently large. But care must be taken for small, non-normal datasets to use appropriate alternate methods.

Copy

**E**

Ok thanks! So if I use nested cross validation (nCV) with N rounds to compare two ML algorithms and for each one I het M\_i and SEM\_i (i in 1,2) to be the estimated mean and SEM respectively (over the N rounds) which one is the best?

When comparing two ML algorithms using nested cross-validation (nCV) with N rounds, and you have the estimated mean performance (M\_i) and the standard error of the mean (SEM\_i) for each algorithm i=1,2, here's how you can determine which one is better:

1. Compare the mean performances M\_1 and M\_2:
   * If the means are significantly different (e.g. M\_1 > M\_2 by a large margin), then the algorithm with the higher mean (M\_1 in this case) is potentially the better performing one.
2. However, the means alone don't tell the full story. You need to account for the uncertainty represented by the SEMs. To do this:
   * Calculate the 95% confidence interval for each algorithm: CI\_1 = M\_1 +/- 1.96 \* SEM\_1 CI\_2 = M\_2 +/- 1.96 \* SEM\_2
   * If the confidence intervals CI\_1 and CI\_2 do not overlap, then you can be 95% confident that the algorithm with the higher mean is truly the better one.
   * If the confidence intervals overlap considerably, then the performance difference between the two algorithms is not statistically significant at the 95% level. You cannot confidently say which one is better.
3. Additionally, you can directly compare the SEM\_1 and SEM\_2 values:
   * A smaller SEM indicates that the mean estimate has higher precision/lower variance.
   * So if M\_1 > M\_2, and SEM\_1 < SEM\_2, it strengthens the evidence that algorithm 1 is better than 2.

In summary, to determine the better algorithm, first compare the means. Then calculate confidence intervals using the SEMs - non-overlapping intervals indicate a statistically significant difference. Also consider the relative magnitudes of the SEMs themselves when interpretating the results.

The SEM quantifies the uncertainty in the mean estimates from the nCV, allowing rigorous statistical comparisons between the algorithm performances.

Copy

**E**

Is it also correct to adjust the confidence level (reduce e.g. 95% to say 70%) until the two C.I's do not overlap and declare the "winner" wthis way with a lower confidence?

Yes, it is technically correct to adjust the confidence level until the confidence intervals (CIs) for the two algorithms do not overlap, and then declare the one with the higher mean as the "winner" at that lower confidence level. However, this practice is generally not recommended in machine learning evaluation or statistical hypothesis testing. Here are a few important considerations:

1. Conventional significance levels (95% or 99% CI) are well-established standards that provide strong evidence against the null hypothesis of no difference between algorithms. Reducing the confidence level increases the probability of making a Type I error (false positive).
2. If you keep adjusting the CI level down until you get non-overlapping intervals, you are effectively increasing the Type I error rate to an unknown, inflated level that is no longer properly controlled.
3. This violates the principle of setting and maintaining a fixed significance threshold a priori before analyzing the data. Adjusting the threshold adaptively based on the observed data is considered "p-hacking" or "data dredging".
4. It makes the analysis highly susceptible to finding spurious effects, especially if multiple algorithms are being compared.

Instead, the recommended approaches are:

1. If the 95% CIs overlap substantially, objectively conclude that there is no statistically significant difference between the algorithms at the 5% level based on your evaluation data.
2. If higher confidence is required, increase your sample size (e.g. more cross-validation rounds) to reduce the SEM and obtain tighter confidence intervals.
3. Consider additional performance metrics or qualitative factors to make a more informed decision between comparable algorithms.
4. In publications, report the precise p-value or CI obtained, allowing readers to judge significance themselves.

So while adjusting CIs until non-overlap is technically possible, it violates statistical best practices and can lead to overconfident conclusions. Maintaining rigorous standards around significance levels is crucial for reliable machine learning model evaluation and selection.

Copy

**E**

You are mentioning p-values in your response above. How they are related to the CI in the context of our discussion (nCV) ?

You're right, I brought up p-values in relation to confidence intervals (CIs) for nested cross-validation (nCV). Let me clarify how they are connected:

In the nCV context, we are essentially performing a hypothesis test by comparing the performance of two algorithms over multiple rounds. The null hypothesis is that there is no difference between their true performance.

The p-value represents the probability of obtaining performance differences as extreme as the observed ones, if the null hypothesis were true. It quantifies how likely the observed results are under the null.

The confidence interval approach and the p-value approach are just two different ways of performing the same hypothesis test:

1. Confidence Interval (CI) Approach:

* We calculate the mean performance and SEM for each algorithm over the nCV rounds
* We construct 95% CIs around each mean using: Mean +/- 1.96 \* SEM
* If the 95% CIs do not overlap, we reject the null hypothesis at 5% significance
* Non-overlapping intervals imply the performance difference is statistically significant

1. P-Value Approach:

* We perform a statistical test (e.g. t-test, ANOVA) on the performance values from nCV
* This test returns a p-value quantifying the chance of seeing results this extreme if null were true
* If p-value < 0.05, we reject the null at 5% significance level
* Small p-value suggests the observed performance difference is statistically significant

So in essence, a 95% CI not containing 0 (for the difference) is equivalent to obtaining a p-value < 0.05 for the corresponding hypothesis test.

The CI has the added benefit of not only testing, but also providing a range that is likely to contain the true performance difference.

Both methods allow you to quantify and interpret the statistical significance of any observed differences between algorithms after nCV evaluation. The choice depends on which form of result is more intuitive for comprehension.

Copy

**E**

So if the algorithms to compare are two and the rounds N how you perform the t-test or ANOVA test? You create two vectors of the performance metric where you store the metrics ordered by the rounds ? Is this a "paired" t-test?

Yes, that's correct. To perform a t-test or ANOVA for comparing two algorithms over N rounds of nested cross-validation, you would set it up as follows:

1. Let's say the performance metric you are tracking is accuracy.
2. For algorithm 1, you will have N accuracy values, one from each round of the outer cross-validation loop. Call this vector: alg1\_accs = [acc\_11, acc\_12, acc\_13, ... acc\_1N]
3. Similarly for algorithm 2: alg2\_accs = [acc\_21, acc\_22, acc\_23, ... acc\_2N]
4. Since these N values come from the exact same outer loop splits, the test will be a paired test, accounting for the pairing/correlation.
5. For a paired t-test:
   * You calculate the paired differences: diffs = alg1\_accs - alg2\_accs
   * This gives you a vector of N difference values
   * You perform a one-sample t-test on these differences
   * The null hypothesis is that the mean(diffs) = 0
   * The alternative hypothesis is mean(diffs) != 0
   * This directly tests if algorithm 1 is better than 2
6. For ANOVA, it would be a repeated measures ANOVA:
   * The Subject factor has N levels (rounds)
   * The Treatment factor has 2 levels (algorithms)
   * You test for a significant Treatment effect while accounting for Subject pairings

Both analyses leverage the pairing in the N rounds to account for variance from the data resampling in cross-validation when testing for performance differences.

The p-value from the paired t-test (or Treatment effect F-test in ANOVA) allows you to interpret if the observed difference is statistically significant at the set alpha level.

This is the rigorous way to statistically compare two algorithms using their performance values from all N rounds of nested CV evaluation.

Copy

**E**

What if the algorithms compared are more than 2 and want to find the best, can you still do the same ?

Yes, if you have more than two algorithms that you want to compare and identify the best performer, you can still use the same overall approach with nested cross-validation and appropriate statistical testing. Here are the steps:

1. Run the nested CV with N outer loop rounds for all K algorithms you want to compare. This will give you K vectors of N performance values each.
2. For pairwise comparisons between any two algorithms i and j:
   * You can use a paired t-test or sign test on the paired difference vector: diffs = alg\_i\_perf - alg\_j\_perf
3. For an overall comparison of all K algorithms simultaneously:
   * You can use a repeated measures ANOVA with:
     + Subject factor having N levels (outer CV rounds)
     + Treatment factor having K levels (algorithms)
   * The ANOVA will test if there is a statistically significant difference in mean performance across the K algorithms
4. If the overall ANOVA finds a significant difference:
   * You can use post-hoc tests (e.g. Tukey's HSD) to identify which specific algorithm(s) differ significantly while controlling the family-wise error rate.
5. You can also construct confidence intervals (e.g. 95% CI) around the mean performance of each algorithm using its SEM over the N rounds.
   * Algorithms whose CIs do not overlap are considered significantly different.
6. To identify the "best" algorithm with high confidence:
   * It should have the highest sample mean performance
   * And its CI should not overlap with the next highest algorithm's CI

This controls the overall Type 1 error rate while allowing you to perform multiple pairwise comparisons and rank the algorithms by their mean performances.

The key things are using an appropriate method (ANOVA or Friedman's test) for the overall omnibus test, followed by post-hoc comparisons or CI visualization for identifying the best performer(s) while accounting for multiplicity.

Copy

**E**

After nCV we usually plot boxplots of the performance metric and compare the medians as the mean is sensitive to outliers (optimistic or pessimistic outer loop splits in our case). Can we use the median and interquantile range in a rigorous statistical way as the above (using CIs) to declare which algorithm is the "best" (assuming we compare two algorithms).

Yes, you can absolutely use the median and interquartile range (IQR) in a rigorous statistical way to compare and identify the best algorithm between two alternatives after nested cross-validation (nCV), instead of relying solely on the means and confidence intervals.

The motivation to use medians and IQRs, as you pointed out, is that they are more robust to outliers compared to means and standard deviations. This is important when some outer loop splits in nCV produce overly optimistic or pessimistic performance values.

Here's how you can proceed with a median/IQR based analysis:

1. After nCV with N outer rounds, calculate the median and IQR separately for each algorithm over their N performance values.
2. Construct median confidence intervals (CIs) around the medians using appropriate methods like bootstrapping or the sign test. The CI should capture the uncertainty in estimating the true median performance.
3. Compare the median CIs:
   * If the CIs do not overlap, you can conclude with high confidence that the algorithm with the higher median is truly better than the other.
   * Report the median value +/- half the IQR range to summarize the overall distribution robustly.
4. Alternatively, you can perform a non-parametric test like the Wilcoxon Signed-Rank test on the paired differences in performance over the N rounds.
   * This tests if the median of the differences is significantly different from 0.
   * The p-value indicates if one algorithm's median performance is significantly higher.
5. You can also visualize the kernel density estimates or box plots for the two algorithm distributions to identify clear shifts and uncertainty.

The key advantages of this median/IQR approach are:

1. It makes no assumptions about the underlying distribution of performances.
2. It is robust to outliers from volatile outer loop splits.
3. With bootstrapping, you can obtain accurate non-parametric CIs around medians.
4. The Wilcoxon test is a non-parametric paired test robust to outliers.

So in cases where algorithm performances violate normality assumptions or have influential outliers, using median statistics with bootstrapping/Wilcoxon tests provides a rigorous non-parametric way to confidently identify the best performer after nCV evaluation.

Copy

**E**

How we can construct median CI intervals with bootstrapping in this case? Do we need to run more nCV rounds or not for that purpose? I am not familiar with this process.

Constructing median confidence intervals (CIs) using bootstrapping in the context of nested cross-validation (nCV) does not require running additional nCV rounds. You can use the existing N performance values from the original nCV analysis.

Here are the steps to bootstrap median CIs for algorithm comparison:

1. Let's say for algorithm A, you have the vector of N performance values from nCV: A\_perfs = [a1, a2, a3, ...., aN]
2. From this vector, compute the sample median: med\_A = median(A\_perfs)
3. Now, generate B bootstrap samples (with B being a large number, say 10,000):
   * For each bootstrap sample b: a) Resample N values with replacement from A\_perfs b) Calculate median of this resampled vector: med\_A\_b
4. You now have B bootstrapped median values: [med\_A\_1, med\_A\_2, ..., med\_A\_B]
5. From this bootstrap distribution, you can calculate quantiles to get a CI around med\_A. Common choices:
   * For 95% CI: [2.5th percentile, 97.5th percentile]
   * For 99% CI: [0.5th percentile, 99.5th percentile]
6. Repeat steps 1-5 for algorithm B to get bootstrapped median CI.
7. If the bootstrapped median CIs for A and B do not overlap, you can conclude their true median performances are significantly different.

The key advantages of bootstrapping:

1. It makes no distributional assumptions about the performance values
2. It accounts for the uncertainty in the median estimate
3. With a large B, the bootstrapped CI will be an accurate representation

No additional nCV runs are needed, as you are resampling from the original N values repeatedly to construct the bootstrap distribution of median estimates.

You can then compare the bootstrapped median CIs to rigorously determine if one algorithm has a significantly higher median performance than the other after accounting for uncertainties.

Copy

**E**

Great! How that generalizes to more than two algorithms to compare using nCV?

The bootstrapping approach to construct median confidence intervals (CIs) and compare algorithms based on their median performances can be extended to more than two algorithms evaluated using nested cross-validation (nCV). Here are the key steps:

1. Let's say you have K algorithms, and for each algorithm i (i=1..K), you obtained N performance values from the N rounds of nCV: alg\_i\_perfs = [p\_i1, p\_i2, p\_i3, ..., p\_iN]
2. For each algorithm i: a) Calculate the sample median: med\_i = median(alg\_i\_perfs) b) Use bootstrapping with B resamples to get B bootstrapped medians: [med\_i1, med\_i2, ..., med\_iB] c) From the B resampled medians, calculate quantile-based CI, e.g. 95% CI
3. You now have K median estimates and their 95% CIs: (med\_1, CI\_1), (med\_2, CI\_2), ..., (med\_K, CI\_K)
4. Visual comparison:
   * Plot the K median estimates with their CIs as error bars
   * Non-overlapping CIs indicate a significant difference in true medians
5. Pairwise tests:
   * For any pair (i, j), check if their median CIs overlap
   * If not, conclude algorithm i has significantly higher/lower median than j
6. Overall comparison:
   * Use a non-parametric test like Friedman's test
   * It tests if the distributions of the K groups (algorithms) differ
   * If significant, follow up with post-hoc tests

The bootstrapped median CIs allow a robust visual comparison that accounts for uncertainty. Pairwise checks identify significantly better/worse algorithms. Overall tests detect if any algorithm is truly different.

A few advantages over using means:

* More robust to outliers from volatile nCV rounds
* Makes no distributional assumptions
* Bootstrapping provides accurate non-parametric CIs

So in summary, by bootstrapping median CIs for each algorithm, you can rigorously identify the best performer(s) based on their median nCV performances, while handling multiple algorithm comparisons in a statistically sound manner.

Copy

**E**

If at the end of the day no algorithm has a clear advantage using the (med\_i, CI\_i) plot comparison is it justified to use the one with the largest median?

If after constructing the bootstrapped median confidence intervals (CIs) for all the algorithms and visualizing them, none of the CIs are clearly separable (i.e. they overlap considerably), then it is generally not justified to simply choose the algorithm with the largest sample median as the "best" performer.

There are a few important reasons why this would be ill-advised:

1. Overlapping CIs indicate that the differences between the true median performances of the algorithms are not statistically significant at the specified confidence level (e.g. 95%). Choosing one just because it has the highest sample median value ignores this uncertainty.
2. The sample medians are estimates that have their own variance/uncertainty. An algorithm with a slightly higher sample median may not actually have a truly higher population median performance.
3. Selecting an algorithm based solely on having the highest sample statistic, despite substantial overlap in CIs, increases your chances of making a Type I error (false positive) in claiming a performance difference.

Instead, if the CIs overlap, the appropriate interpretations are:

1. Insufficient evidence to conclusively differentiate the algorithms based on their median performances using the evaluation data at hand.
2. The algorithms likely have very similar or essentially indistinguishable true median performances within the resolution of your evaluation procedure.
3. Consider them as statistically comparable/equivalent options in terms of median performance.

Potential next steps could be:

1. Gather more data (e.g. increase number of CV rounds) to reduce uncertainty and obtain tighter CIs for better resolution.
2. Examine other performance metrics or qualitative factors to make a more informed selection.
3. Use alternate statistical tests (e.g. non-parametric tests) to check if any performance distributions differ.
4. In the absence of a clear winner, strategically select the simplest or most pragmatic algorithm as an objective tiebreaker.

But arbitrarily choosing one algorithm simply because it has the highest sample median value, when there is substantial overlap in the CIs, goes against statistical best practices and risks making an unjustified choice. The uncertainty indicated by the CIs must be accounted for responsibly.