Lab Insight

Performance at Scale - Comparing AI/ML Performance of SAS Viya vs. Alternatives

Authored by:

Russ Fellows

June - 2023

Overview

There are several important trends for businesses today including cloud computing, the move to container native applications and the use of Artificial Intelligence. SAS® Viya® sits at the intersection of all three trends by offering an advanced AI solution that can operate in both public and private cloud environments running as a containerized cloud native application.

With the move to cloud computing, companies gain operational flexibility that enables them to optimize their workloads across a hybrid operating environment, including both on premises and public cloud environments. Recently, the use of Artificial Intelligence (AI) has been growing exponentially within companies, and in particular machine learning (AI/ML) has proven to be particularly valuable, enabling companies to find and exploit insights hidden within their data.

Understanding the scalability, relative performance and cost effectiveness of AI solutions is an important aspect when choosing a platform for use within a company. In particular, the speed or performance of an AI platform is critical for many reasons, due to the operational cost implications and its ability to support running models at scale with increasing data set sizes. Note: Information about the data used for testing is provided in the Appendix.

SAS asked The Futurum Group to help test, analyze and summarize the results of a competitive test comparing several leading AI/ML options. All testing was performed in the Azure public cloud, using multiple instance sizes to showcase the scalability and effectiveness of each AI/ML platform. The Futurum Group Labs worked with SAS to establish the test objectives, along with the test process and data sets utilized. The findings and results presented were independently created by Futurum Group Labs from the testing process. For details of the calculations below, see Appendix A. The findings include:

- SAS Viya is on average 30X faster than all competitors tested across all test cases
- Up to 326X faster and average of 49X faster than "Competitor-A" running in Azure
- Viya produced the same results for 86%+ lower cost vs. competitors overall

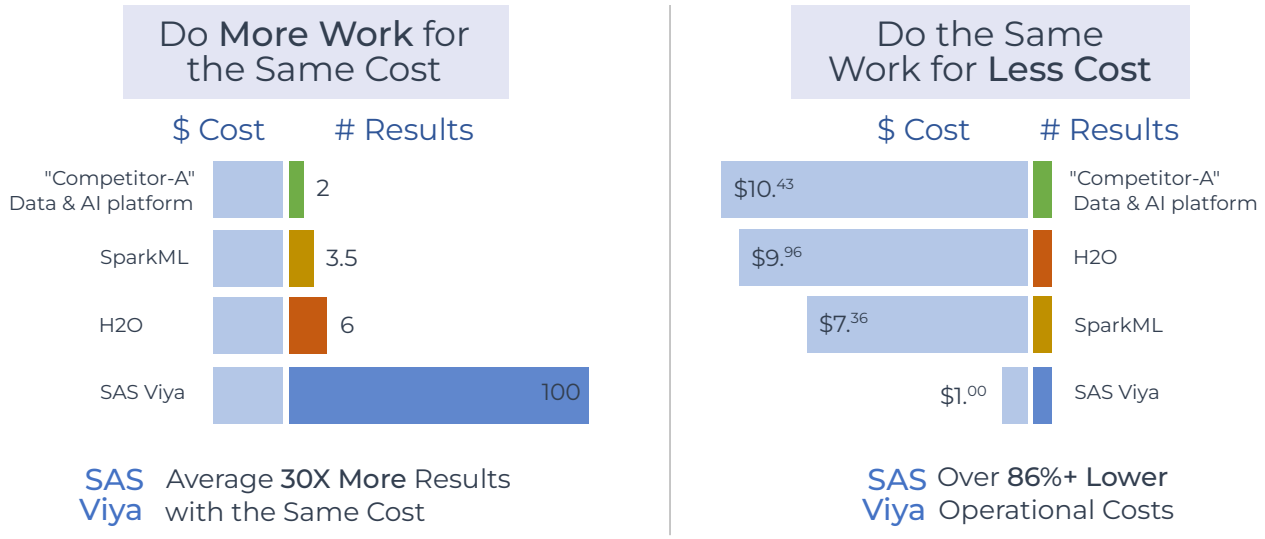


Figure 1: SAS Viya Performance Impact (Source: The Futurum Group)

SAS Viya Highlights

SAS Viya is a cloud-native AI, Analytics and Data Management Platform that enables businesses to leverage data to gain insights, then make predictions and decisions. SAS Viya is designed for a variety of users including Business Analysts, Data Scientists, Data Engineers, MLOps Engineers, or anyone tasked with turning data into decisions. SAS Viya's integration and automation capabilities enable decision making through data exploration & visualization, model development & deployment, and Intelligent decisioning.

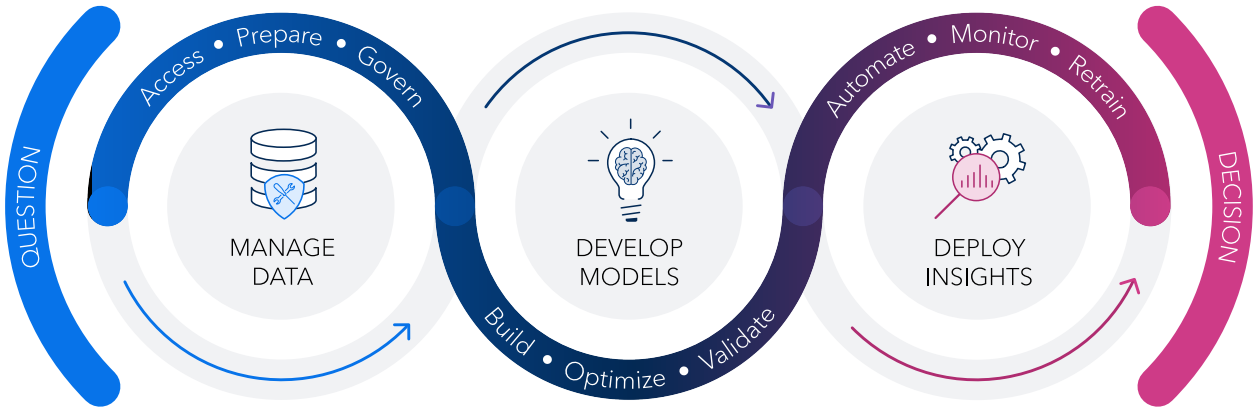


Figure 2: AI & Analytics Lifecycle in SAS Viya (Source: SAS)

AI/ML Platform Requirements

The use cases for Artificial Intelligence are growing daily, with generative AI and other types of AI being deployed for new business problems. In particular, machine learning (AI/ML) has proven to be particularly effective for gaining insights from data and then applying that knowledge to solve business problems.

Looking specifically at AI/ML, four machine learning models are commonly used for predictive modeling and decision-making purposes: Linear Regression, Logistic Regression, Random Forest, and Gradient Boosting. Additionally, ensemble models and variations of these models could be used to enhance the accuracy of single algorithms.

There are many requirements for an AI/ML platform, including the ability to rapidly ingest, refine and examine data for errors and missing values. Additionally, prototyping multiple AI/ML models to evaluate alternatives using automated "tournament" capabilities can be beneficial. All these steps may be accomplished more efficiently if the AI/ML platform provides a user interface (UI), including automation, embedded best practices, and low-code / no-code capabilities.

Some of the most important considerations when choosing an AI/ML platform are its support for multiple types of AI models, the flexibility to run in public clouds and on premises, along with a graphical UI and other ease of use features. Finally, performance and scalability are critically important considerations.

SAS Viya is compared to "Competitor-A", a commercial Data + AI platform, two open-source general purpose platform and several special purpose libraries. For each of the desired features, we compare SAS Viya to the competing options tested, with the results shown in Table 1 below.

Capability	SAS® Viya	Competitor-A	Other General	Specialty Library
Broad ML Support: Supports Multiple ML Types	Yes	Yes	Yes	No: One type only
Runtime Flexibility: On-Prem and Cloud Options	Yes	No: Cloud only	Some: Limited options	Yes
Scalability: Scales to 100's M Data Elements	Yes	No: Failed to complete	Limited: Some models failed to complete	Some: Scales for specific ML types only
Ease of Use: Graphical UI w/ no-code	Yes	Limited: UI lacks no-code	Limited: No UI, lacks no-code	No
Performance: Relative Performance	Best General Library	Third out of four	Second and fourth of four	Good: specific ML types only

Table 1: AI/ML Features & Capability Comparison (Source: The Futurum Group)

Test Process Overview

The Futurum Group Labs worked with SAS to establish the test objectives, along with the test process and data sets utilized. Relevant performance data and other metrics were provided to Futurum Group Labs for review and analysis. The findings and results were independently created by Futurum Group Labs from the testing process and results obtained. The testing process consisted of the following:

- Four AI/ML Model Types
  - Gradient Boosting, Random Forest, Linear Regression, and Logistic Regression
- SAS Viya vs. general-purpose and specialty libraries
  - Spark (version 3.3.2), H2O-3 (3.40.0.2), Competitor-A (latest version on: March 2023)
  - Gradient Boosting specialist libraries: LightGBM (vers. 3.3.5), XGBoost (vers. 1.7.4)
  - Random Forest specialist library: Ranger (version 0.14.2)
- Over 1,500 total tests
  - 100 configurations & multiple runs for each combination

All testing was conducted within Microsoft Azure, using the Azure Kubernetes Service (AKS). All instances other than Competitor-A ran in a cluster which was deployed using SAS' Infrastructure as Code (IaC) project available on GitHub. For Competitor-A, all datasets were loaded into their proprietary environment, and run within the service offering. The node sizes chosen for execution were identical to those used for testing of Viya, other general and specialty libraries.

SAS Viya was installed using the SAS Viya Deployment project and the version of SAS Viya installed was 2023.02. The team used the "minimal" deployment configuration from the IaC SAS project, and the initial

deployment of the infrastructure was made with two modifications; first to set the minimum and maximum number of nodes to one (1), with the second modification to change the default instance size of the node pool virtual machine (VM) to the desired size for each test.

All the programs, except those for Competitor-A, were executed as Kubernetes jobs on a custom container image which contained the required software (Python and R with the required packages pre-installed). The jobs were set to execute on only the CAS node pool of the cluster ensuring that each program executed on the same machine with access to the same resources. Each program was executed multiple times[1] and the numbers presented are an average of the execution times of the executions.

When all the different combinations of library, algorithm, and dataset had been executed on that instance size, the CAS node pool was deleted and recreated with a new underlying instance size: 16, 32, 64, and 72 vCPU instances. Details of the specific instances used for the environment can be found in the Appendix.

The data sets for all the programs, apart for Competitor-A, were stored on a Network File Share (NFS) VM within the same resource group as the AKS cluster. The data was pre-loaded into the environment before the test programs were executed. The NFS share was mounted to the container as part of the Kubernetes job definition.

Shown below in Figure 3 is a high-level diagram of the Azure test environment, depicting container images in ACR registry, executing in AKS, and using data from an Azure VM running as an NFS server for access to all ML model libraries and datasets.

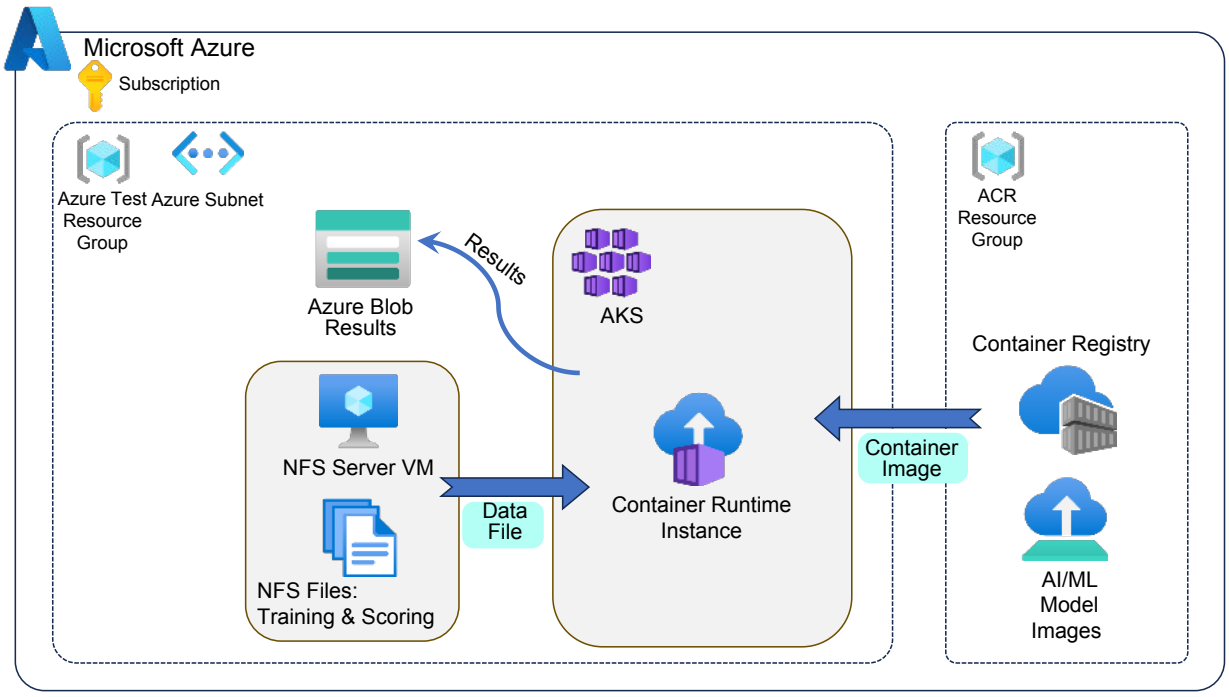


Figure 3: Diagram of test environment (Source: SAS & Futurum Group)

[1] The programs were executed 3 times apart from where a program failed the first two times in which case a third attempt was not made.

For Competitor-A, the environment was configured as a single node cluster with the programs executed manually from within the platform's user interface. The datasets for Competitor-A were stored in Azure Storage. When all the different combinations of algorithm and dataset had been executed, the worker node was deleted and recreated with a new underlying instance size. Details of the instances used can be found in the Appendix.

All data sets were cleaned and checked for consistency, errors and missing values prior to using them for training and scoring. Moreover, while SAS Viya has tools and features designed to assist with these steps, the potential advantages of SAS Viya were not part of testing.

The steps performed included the following for each of the libraries tested, and for each test case:

1. Start Analytics Engine (SAS/CAS, H2O, Spark, etc.)
2. Load Training and Test data
3. Pre-process data (Note: Minimal pre-processing was run to enable all algorithms to run without failures, eg. Imputation).
4. Train model
5. Score model
6. Calculate Accuracy and Finalize Results

Competitive Test Results

As outlined previously, all testing was performed in Azure, using the AKS service to run each of the AI/ML models for each ML type and for each dataset utilized. Some of the highlights observed during testing include the following:

- SAS Viya outperformed other general-purpose libraries while using 87% less CPU resources across a majority of the 20 different datasets tested
- Of the 50 tested combinations for linear regression and logistic regression, SAS Viya produced the results faster in 49 out of the 50 tested configurations.
- For large or complex datasets with a high number of features (columns), Viya's efficiency and performance advantages increased, providing over a 30X advantage on average compared to other general-purpose libraries.
- While Viya completed all but one of the 100 configurations, several competing general-purpose libraries failed multiple complex datasets using any instance size.

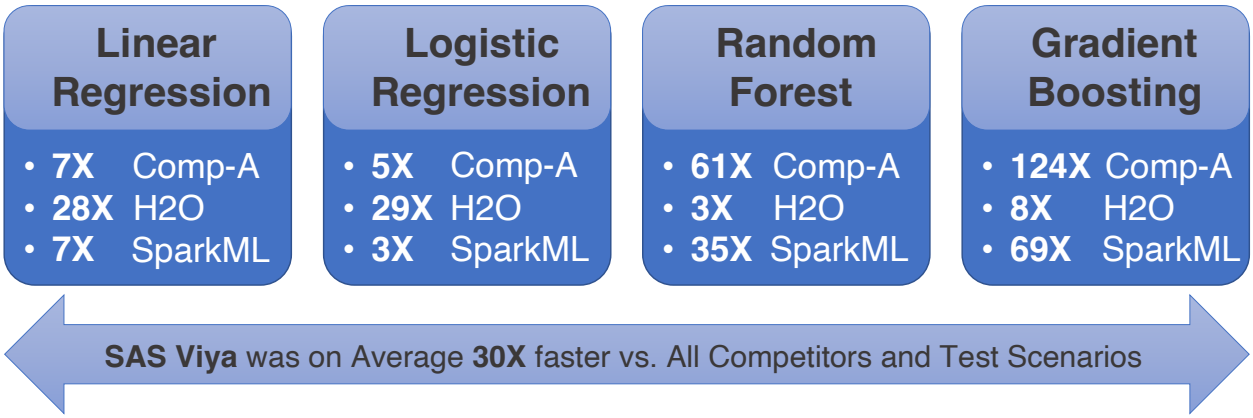


Figure 4: Summary of SAS Viya vs. Competitors (Source: The Futurum Group)

Futurum Group Comment: *The comparative results for SAS Viya are overwhelmingly positive. Although there were instances where SAS was outperformed by special purpose libraries, Viya's ability to outperform other general-purpose libraries for 97% of test cases is quite exceptional.*

The fact that SAS Viya was significantly faster than competitors across nearly every configuration with general purpose libraries has many practical and cost advantages, enabling substantial flexibility and operational savings.

As a result, a Data Scientist or AI Practitioner can run a "tournament" to help identify the best model options for a given dataset with a small size instance. In runtime environments where time and instance size are directly correlated to cost, Viya's ability to produce the same results in less time while using a smaller instance has significant cost saving implications.

With 100 different test configurations tested for each library, we summarize some of the most interesting and important results, leaving the remaining details for the Appendix. Although SAS Viya did not win every

configuration, it was either the best, or among the best for every test conducted. The details for testing are provided in Appendix A, with the results for all 100 configurations for each library contained in Appendix B.

Linear Regression

For the 100 test cases run for Linear Regression, each of the 4 libraries was tested using 5 different data sets and 5 different configurations, using "ordinary" regression, rather than penalized regression. SAS Viya outperformed the competitors in every test case, and in many cases the smallest 8 CPU instances outperformed larger configurations from competitors.

Looking at a relatively simple, but large dataset with 10 features / columns, containing 100 million observations / rows of data. This example shows one of the closest comparisons for SAS Viya, although SAS Viya is faster and shows superior scalability, by the fact that with increasing CPU capacity Viya's time decreases relative to other competitors.

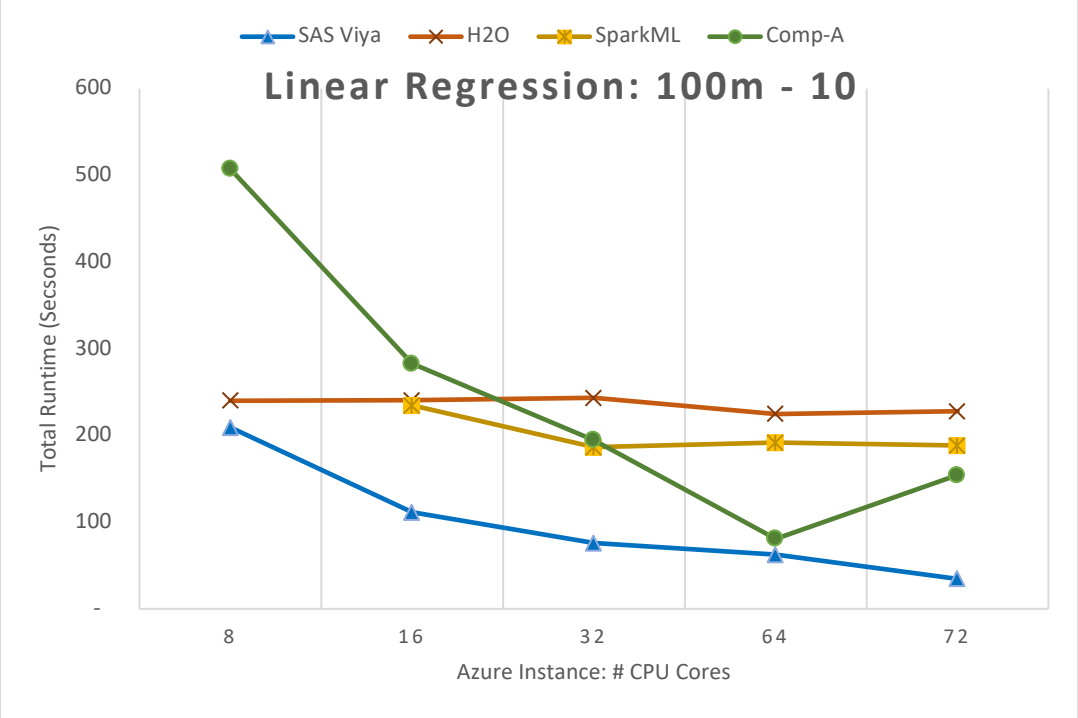


Figure 5: Linear Regression Test, 100m-10 (Source: The Futurum Group)

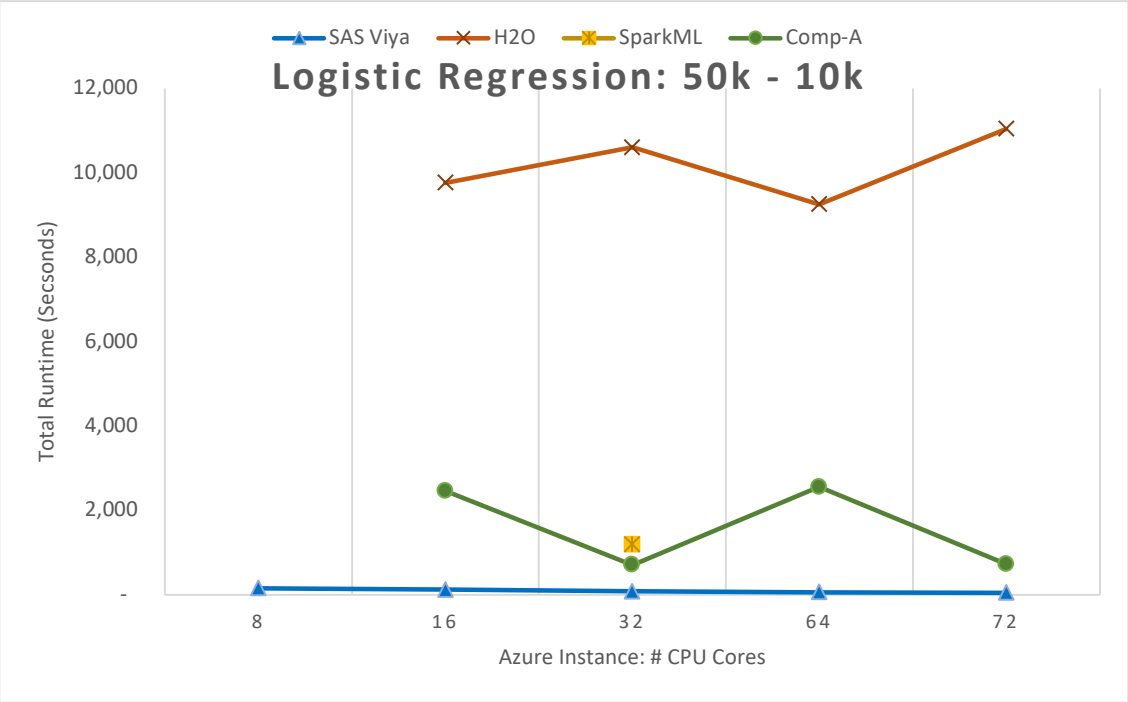
The dataset shown above had 10 features and although there were 100m observations, each of the libraries handled this case relatively well. This example shows one of the closest results, although SAS Viya still outperformed every competitor, with the 16 CPU size nearly matching the largest instance sizes from the remaining competitors. All linear regression test results are provided in Appendix B, along with a table showing the results numerically.

***Futurum Group Comment**: Even for data sets with few features and hence low complexity, SAS Viya still outperformed the competition at every size instance runtime size. Additionally, SAS Viya's performance scaled nearly linearly with the addition of CPU and memory available in the larger instance sizes.*

Logistic Regression

For the 100 test cases run for Logistic Regression, each of the 4 libraries was tested using 5 different data sets and 5 different configurations, using "ordinary" regression, rather than penalized regression. Many of these results were similar in that SAS Viya was consistently better than the three alternatives tested. In particular, "complex" datasets that contained a high number of features, Viya's advantages became very clear increasing its advantage over alternatives by an order of magnitude.

One example that particularly highlights SAS Viya's advantage processing complex datasets is a Logistic Regression test using a dataset with 50K observations (rows) and 10K features (columns). In Figure 6 below, SAS Viya outperformed a competitor by 137X at the largest CPU instance. Also notable is the fact that SparkML failed to complete 4 out of 5 of the configurations. SAS Viya not only outperformed all competitors but running Viya with a small 8 CPU configuration produced results in 77 seconds. By comparison, the next fastest time was Competitor-A running on a 72 CPU instance, which required 567 seconds to produce a similar result. Thus, SAS Viya running on the smallest instance was 7.3X faster than the next closest alternative running on the largest instance size.



- * **Note:** SparkML failed 4 of the 5 tested configurations

Figure 6: Logistic Regression Test, 50k-10k (Source: The Futurum Group)

In this example, SAS Viya was able to complete the same task in under 77 seconds using the smallest, 8 CPU instance size. The next closest performing configuration was Competitor-A using 72 CPU instance, which required 567 seconds. Moreover, Competitor-A's best result used 8X more resources ((72 -8) / 8 = 8) to produce the same results and required more than 7X longer (567 / 77 = 7.3) to do so. Regardless of whether these models are utilizing public cloud or private infrastructure, time and resource consumption have significant business implications.

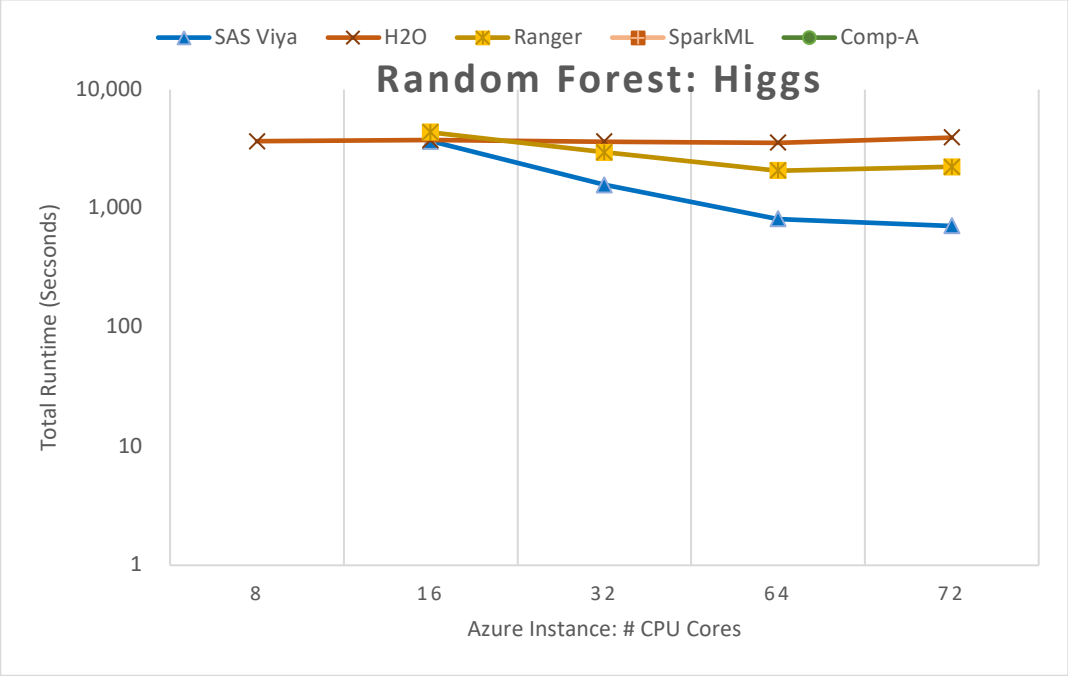
***Futurum Group Comment**: The larger and more complex the data set, the bigger the advantage for Viya. Additionally, SAS was able to produce the same model results orders of magnitude faster, running in seconds or minutes, vs. hours or failing completely for some competing alternatives.*

The example shown above is relatively complex in that there are a large number of "features" or columns, which may be thought of as a wide data set. Although the number of rows, or observations is also important, a large number of features generally resulted in longer run times for all of Viya's competitors. There are four more logistic regression test cases, and in each case, SAS Viya outperformed other options by significant margins. A summary of all test results is provided in Appendix B.

Random Forest Details

In general, Random Forest was the most resource intensive of the four ML model types tested. This is due to how Random Forest inherently facilitates parallelization, which can potentially use a substantial amount of memory when employing multi-threading. As a result, this approach consumes more CPU and memory while processing datasets. For Random Forest models, the "Ranger" special purpose library was used, in addition to the four general-purpose AI/ML libraries tested for all configurations.

The largest dataset tested for both Gradient Boosting and Random Forest was known as the "Higgs" dataset, which contains over 300 million unique data elements, which is the product of 28 features and 10.5 million training observations plus 500 k test observations. Due to the vast differences between the options, the graph uses a logarithmic scale, where a difference that appears to be 2X, equates to a 10x difference.



- * **Note**: SparkML and Competitor-A **failed** all tested configurations

Figure 7: Random Forest: Logarithmic Higgs Dataset Performance (Source: The Futurum Group)

***Futurum Group Comment**: The fact that two of the other general-purpose libraries failed all tested configurations is very significant. Companies using these competitors would be forced to abandon Random Forest as a potential solution, eliminating a potentially superior model option. Once again, a 16 CPU instance of SAS outperforms all competitors at the same or larger instance sizes.*

As seen above in Figure 7 plotted using a logarithmic scale, two of the primary competitors failed to run any of the tested configurations. Specifically, both SparkML and Competitor-A failed all the configuration sizes, multiple times. The only other general-purpose library to complete testing was H2O, which required approximately 1 hour to produce results. The special library Ranger failed on the smallest instance size, as did SAS Viya. However, for 16 CPU's and above, Viya outperformed all competitors, increasing its advantage as the instance size increased.

Gradient Boosting Details

First, we examine a test case where SAS Viya was not the absolute fastest and its advantage over the competition was "only" about 10 – 100X better than other general-purpose libraries. For this test the AI/ML type was Gradient Boosting, with a dataset known as "Comcast". In this specific case, Viya outperformed all three general purpose libraries, H2O, SparkML and Competitor-A.

As previously noted, external libraries such as LightGBM or XGBoost may be used within SAS Viya. Additionally, Viya provides a specific wrapper for LightGBM which enables this library to be utilized rather than the native Gradient Boosting algorithm within SAS Viya if desired. Shown in Figure 8 below is a logarithmic scale graph comparing a Gradient Boosting model for the Comcast dataset, showing the general-purpose libraries along with LightGBM and XGBoost.

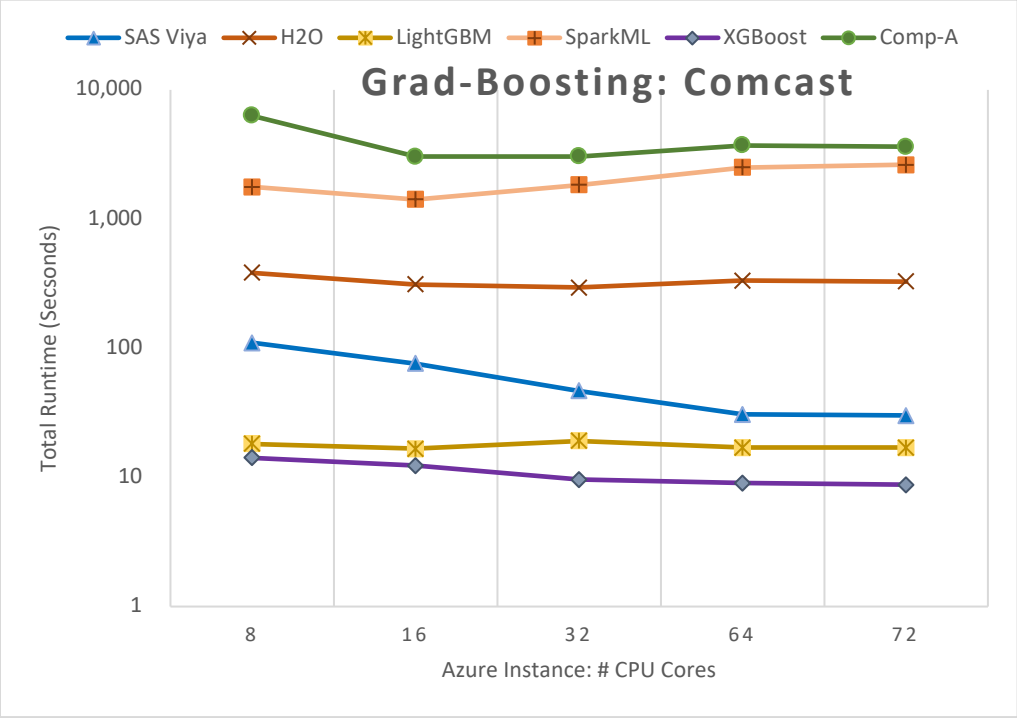


Figure 8: Gradient Boosting: Comcast Logarithmic Performance Chart (Source: The Futurum Group)

Again, due to the differences between the competitors, the graph uses a logarithmic scale. As seen above, SAS Viya was among the leaders, similar to both LightGBM and XGBoost. In contrast, H2O was an order of magnitude slower, and both SparkML and Competitor-A were over 100X slower.

There are two notable aspects of the results above. The first is that SAS Viya demonstrated nearly linear scalability, with each doubling of CPU and memory resulting in a nearly 50% reduction in time. Additionally, as with nearly every example even the smallest CPU instance size of Viya outperformed every size of the competing general-purpose libraries, resulting in the ability to significantly reduce operating costs.

***Futurum Group Comment**: Even in cases where SAS Viya is not the absolute fastest, it is nearly as fast as the special purpose libraries, and 10 – 100x faster than competing general-purpose libraries. Again, the smallest 8 CPU instance of SAS outperforms other general libraries, and larger instance sizes nearly equals the special purpose libraries.*

Performance at Scale

As just shown, SAS Viya's performance advantage increased when training complex datasets with a relatively large number of features or columns. In test cases for each of the four ML model types, those having more features generally resulted in SAS Viya's performance advantage increasing compared to simpler datasets containing fewer columns. The largest dataset tested, known as Higgs, contained 28 columns and 10.5 million observations in the training dataset and 500k observations in the scoring dataset. The total number of unique data elements is the sum of the product (28 * (10.5 + 0.5) m = 308 m), 308 million.

One critical way of evaluating algorithms efficiency is to plot the amount of time required to solve problems relative to other algorithms. The most efficient algorithms are more scalable in that they can process larger datasets significantly faster than less efficient algorithms. At some point or data size, the inefficient algorithms are unable to complete a task in a reasonable time.

In Figure 9 we plot the runtime required to process 5 million data elements for the Comcast dataset, and 308 million data elements from the Higgs dataset using Random Forest model. SparkML was unable to process 5 million data elements in several instances and required up to 20,000 seconds of runtime before failing. While Competitor-A was able to complete processing the Comcast 5m datapoint, it failed when attempting to process the 308 m data elements contained in the Higgs dataset, again after running for up to 20,000 seconds. Meanwhile, H2O, Ranger and SAS Viya all completed both Comcast and Higgs datasets. Notably SAS Viya outperformed all competitors on the Higgs dataset, indicating it is the most efficient.

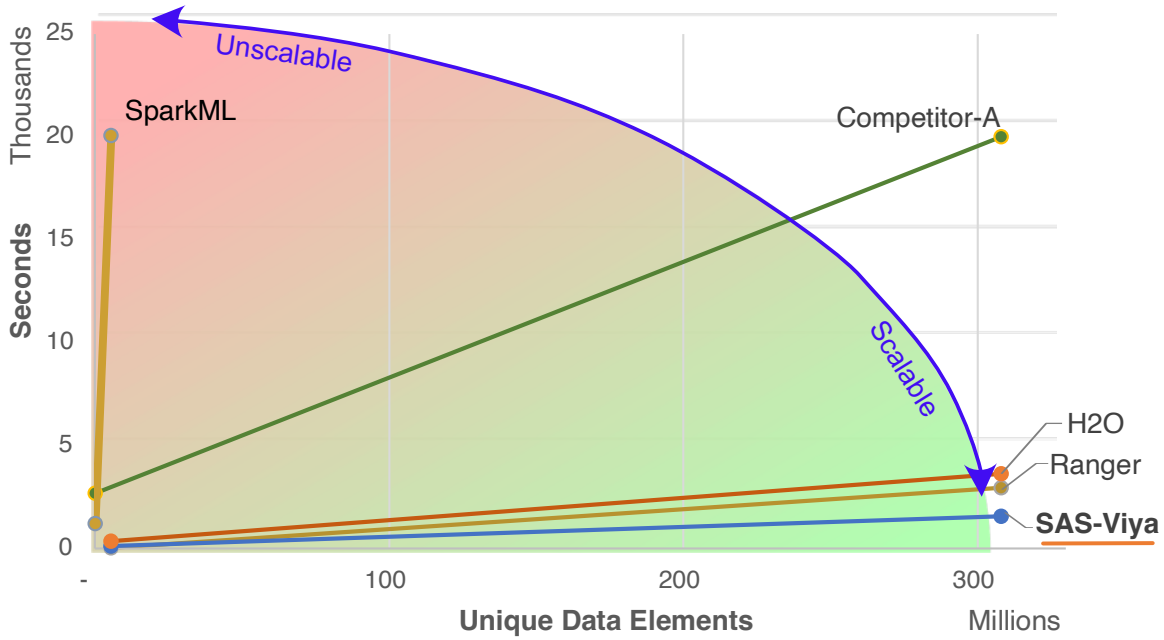


Figure 9: Library Efficiency Comparison, Random Forest Higgs dataset (Source: The Futurum Group)

For the "Higgs" dataset, SparkML and Competitor-A failed at every instance size when training a Random Forest model, SAS Viya was able to complete the training with 16 CPU instance size and above. Given that both SparkML and Competitor-A failed, their run time was reported as the maximum time allowed by our testing, 20,000 seconds.

***Futurum Group Comment**: Companies using more efficient tools have a significant competitive advantage over their peers utilizing inferior tools. Data Scientists and businesses have found that the accuracy and relevance of AI/ML models is directly correlated with the size of training data. The ability to effectively process ever larger datasets is dependent upon the efficiency and scalability of the tools utilized.*

Investigating this example further, we then evaluated the scaling ability of the three successful algorithms, SAS-Viya, Ranger and H2O. We found that Viya was able to scale performance nearly linearly with additional CPU power, decreasing runtime from 3,662 seconds with 16 CPUs to 1,576 with 32, and down to 815 seconds with 64 CPUs. By comparison, both H2O and the special library Ranger showed almost no improvement or decrease in runtime when using more than 32 CPUs.

In Figure 10 below we plot the time reduction moving from a 32 CPU instance to a 64 CPU instance and show the percentage of time decrease. Ideally, the reduction would be 50%, due to a 2X increase in CPU resources.

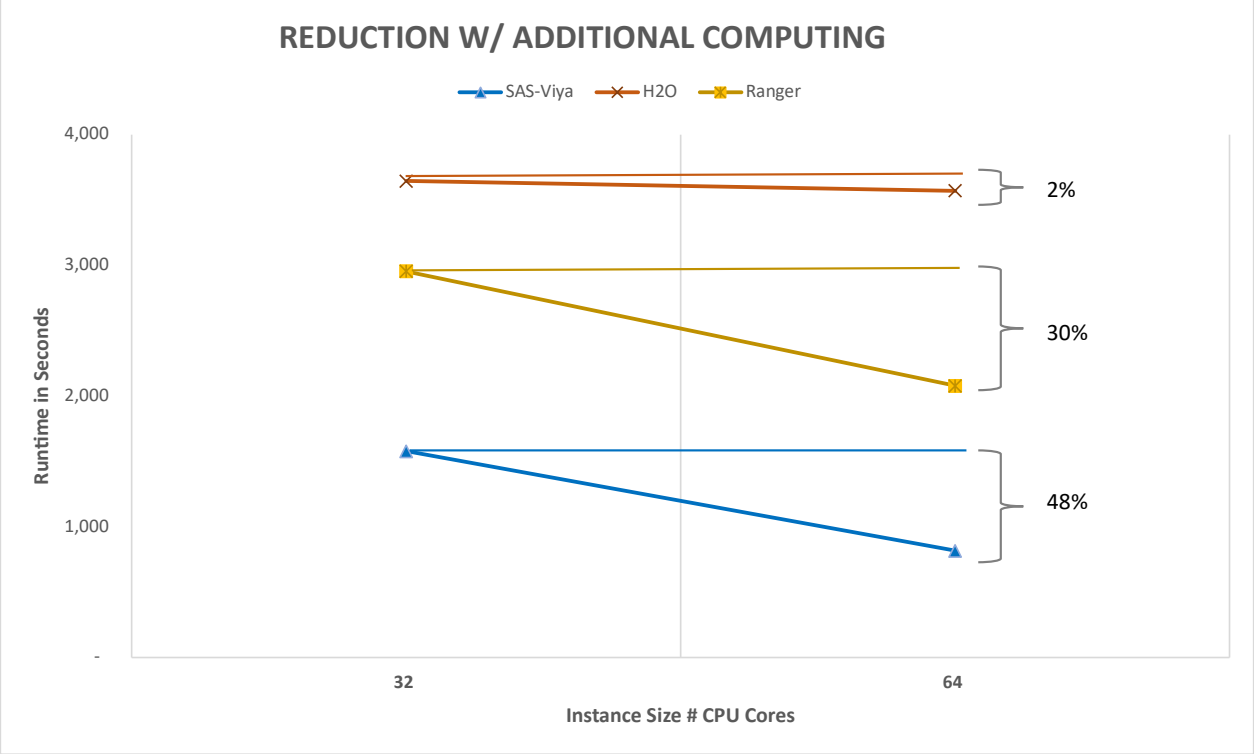


Figure 10: Library Scalability Comparison, Random Forest Higgs dataset (Source: The Futurum Group)

As seen above, SAS Viya scaled nearly linearly with an increase in resources. Reducing runtime by 48% achieved the nearly ideal 50% reduction. In comparison, the Ranger specialty library only achieved a 30% reduction, while H2O had virtually no change decreasing runtime by only 2%.

The specific values are shown in Table 2.

Runtime in Seconds			% Benefit
CPU Cores	32	64	50%
SAS-Viya	1,576.69	815.58	48%
H2O	3,643.83	3,569.43	2%
Ranger	2,953.99	2,077.78	30%

Table 2: Runtime vs. CPU Cores for Random Forest Higgs (Source: The Futurum Group)

Futurum Group Comment: *SAS Viya's scaling from 32 to 64 CPU cores was excellent, achieving a nearly linear reduction of 48%. Meanwhile other general-purpose libraries struggled to complete the Higgs dataset or were unable to scale and the specialty Ranger library had lower scalability and longer run times than Viya.*

Final Thoughts

With companies increasingly consuming IT resources as a service, businesses have direct visibility into the cost of running applications. Although performance has always been important, with cloud computing time equates directly to operational expense. The ability to produce desired results with less cost, or produce better results for the same cost provides companies with a significant competitive advantage.

***Futurum Group Comment**: SAS Viya's ability to deliver the same results for 86%+ less cost on average or deliver 30X more results for the same resource utilization provides a high degree of flexibility to companies looking to optimize operational costs and AI/ML results.*

The modern, cloud native design of SAS Viya provides the ability to run across a variety of popular container platforms including generic Kubernetes, OpenShift along with cloud options such as Azure Kubernetes Services (AKS), Amazon's Elastic Kubernetes Service (EKS) and others. This operational flexibility enables businesses to choose the optimal location for their data and AI/ML models, based upon cost, data governance, privacy or other considerations.

Beyond cost, scalability is a critical factor for enabling AI/ML models to process large data sets. Using inefficient AI/ML platforms and algorithms that are unable to support large or complex datasets means eliminating some types of models or datasets from consideration, placing the company at a significant disadvantage relative to its peers using more efficient tools. SAS Viya's efficiency and performance result in its ability to dramatically outperform competing solutions, particularly at scale with large and complex datasets.

***Futurum Group Comment**: Where competing solutions ran for hours, or failed completely, SAS Viya was able to deliver results in minutes. For companies seeking a competitive edge, the ability to produce results on average 30X faster than a competitor provides a massive competitive advantage.*

SAS Viya's performance and scalability advantages alone make Viya a leading AI/ML platform, without considering other features such as Viya's UI, data management, collaboration, deployment and other AI/ML operations. Additionally, the data visualization capabilities help businesses quickly identify trends or outcomes that could otherwise be overlooked.

Working with SAS, Futurum group's testing showed that SAS Viya outpaced competing alternatives, producing models with the same accuracy while running faster and consuming significantly less computational resources. Moreover, SAS Viya should be considered a leading AI/ML platform by companies evaluating how best to leverage AI in a modern, hybrid cloud environment.

Appendix A – Test Environment

The details of the testing, including the test process, the hardware and software environments and other relevant configuration details are provided in section A of the Appendix. The subsequent Appendix B contains the test results for each configuration tested.

Test Process

The test environment utilized a private Azure Container Registry (ACR) to act as a repository for Kubernetes (K8s) container instances. These were deployed using a YAML deployment job to execute each specific test. Both training and testing (aka scoring) data resided on a separate node in NFS mounted repository, with data loaded prior to testing. All test environments utilized the same NFS server for test data, except for Competitor-A, which required its own proprietary data repository. The exact same test files were loaded into the proprietary repository for use by Competitor-A.

Note: A diagram of the test environment was presented previously on page 4 as Figure 3.

The Setup for the test required the following setup:

1. Create Azure container runtime environment in Azure Kubernetes Service (AKS):
   i) The container execution machine instance size is specified at creation time. Thus, the node pool infrastructure was instantiated for each instance size tested
2. Load test data into private NFS Server that is accessible by AKS

The high-level test process occurred as outlined below:

1. Apply a test job into AKS that pulls a container image from the ACR
2. Parse results
3. Collect container logs and result data and store in Azure Blob Storage for each test
4. Collect metric data from SAS Enterprise Session Monitor and store in Azure Blob Storage for each test

Test Environment

Each program was executed as a Kubernetes job in its own container. There were two base container images utilized, one type was used to execute all Python programs, and the other was used for executing all R language-based executions. Each image has either Python or R, plus *all* of the library packages pre-installed along with Java 11.

For containers operating within a Kubernetes environment, configuration files known as YAML (Yet Another Meta Language) are used for specification and control.

A separate YAML file was used to define each test executed, one for each of the 75 different test cases that were run outside of the proprietary environment of Competitor-A,

When these YAML were "applied" via command line "*kubectl*" command to the cluster, a pod with a single container was created. Part of the pod definition includes a mount path to the location where both the test

data, and the program code is stored in the cluster as a K8s configMap. Depending on the library being executed, some required environment variables were set to indicate the instance size, and how many CPU cores a particular test should use. Each of the base images are configured to execute a program at "/root/program.py" or "/root/program.R".

Additionally, the YAML configurations include a tolerations section that ensures that the job only ever executes on the node pool which contains 1 container instance. Each of the workloads runs within the container, with the exception of SAS and Competitor-A. In the case of SAS Viya, the startup container calls to a SAS Viya service to execute SAS on that same node pool instance. For Competitor-A, their model requires execution within the Azure service environment. However, identical instance sizes were chosen for all testing including for SAS Viya and Competitor-A.

Hardware Environment

The hardware utilized for all testing were Azure instances, using both DXs_v3, and FXs_v2, where the "X" refers to the specific number of virtual CPUs (vCPU's) allocated. Each instance size also had a corresponding amount of system memory (DRAM) allocated and varied by CPU size. The specific sizes used for testing were:

- D8s_v3: 8 virtual Intel Broadwell class CPU cores, and 32 GB of DRAM
- D16s_v3: 16 virtual Intel Broadwell class CPU cores, and 64 GB of DRAM
- D32s_v3: 32 virtual Intel Broadwell class CPU cores, and 128 GB of DRAM
- F64s_v2 64 virtual Intel Broadwell class CPU cores, and 128 GB of DRAM
- F72s_v2 72 virtual Intel Broadwell class CPU cores, and 144 GB of DRAM

Accuracy of Results

Accuracy is one critical measurement of the AI/ML process. As previously stated, the resulting accuracy of every model was the same to within approximately 1%. Although there were minor differences, the scoring accuracy was the same to three digits of precision. As the accuracy of each model was statistically the same, the specific accuracy results for each test are not reported.

Analysis of Results

In the subsequent Appendix B, we present all the test results in both tabular and graphical formats. For each test configuration, each step in the testing process was measured, with the sum or total time being reported. For performance, lower or less time being better. The results are all displayed in seconds, which includes the following steps:

1. Start Engine
2. Load Training Data
3. Load Test Data
4. Pre-process data
5. Train Model (using training data)
6. Score Model (against testing data)
7. Get Results and calculate accuracy
8. Stop Engine

Calculations for Work vs. Cost

In Figure 1 on page 1, we presented a graphic with a comparison of the amount of work performed and the cost to perform the work.

Looking first at the left side of the Figure 1, the calculations for the "Amount of Work" completed come from the calculations in Appendix Table 1 below, showing the number of times slower each library is than SAS Viya. Using these results as a percent of 100 shows the relative amount of work vs. SAS Viya. The relative advantages are then calculated as a percent of 100%, with SAS being 100% of results, H2O at 6, SparkML at 3.5 and Competitor-A at 2.

# Times Slower vs. SAS Viya / Test					
Test Cases	SAS-Viya	H2O	SparkML	Comp.-A	Avg. Slower
Gradient Boost	**1.00**	**8.03**	**68.96**	**124.27**	
adult	1.00	3.10	107.10	243.40	117.87
amazon	1.00	4.30	139.50	246.90	130.23
comcast	1.00	7.10	48.60	80.70	45.47
higgs	1.00	24.20	20.90	-	22.55
upsell	1.00	1.50	28.80	51.80	27.37
Linear Regression	**1.00**	**27.47**	**7.36**	**7.42**	
glm-100k-500	1.00	5.50	5.60	1.90	4.33
glm-100m-10	1.00	3.30	2.60	2.60	2.83
glm-10k-2k	1.00	14.50	17.90	8.80	13.73
glm-1m-50	1.00	3.20	5.20	3.00	3.80
glm-50k-10k	1.00	110.90	5.50	20.70	45.70
Logistic Regressior	**1.00**	**29.15**	**3.61**	**5.12**	
log-100k-1k	1.00	12.30	5.10	2.10	6.50
log-100m-10	1.00	2.40	1.50	1.90	1.93
log-1m-50	1.00	3.20	4.60	2.20	3.33
log-500k-500	1.00	10.30	4.00	3.00	5.77
log-50k-10k	1.00	111.80	2.80	16.30	43.63
Random Forest	**1.00**	**3.11**	**34.63**	**60.68**	
adult	1.00	2.90	50.50	74.20	42.53
amazon	1.00	4.20	43.50	98.00	48.57
comcast	1.00	3.00	6.10	57.80	22.30
higgs	1.00	3.30	-	-	3.30
upsell	1.00	2.20	10.70	12.20	8.37
Average Slower	**1.00**	**16.94**	**28.64**	**49.37**	**30.01**
% of 100	**100.00**	**5.90**	**3.49**	**2.03**	

Appendix Table 1:

Times SAS-Viya is faster by library and data set

Test RunTime Costs					
Test Cases	SAS-Viya	H2O	SparkML	Comp.-A	
Gradient Boost	**$ 2.10**	**31.39**	**45.51**	**37.45**	
adult	$ 0.04	0.09	3.62	8.80	
amazon	$ 0.03	0.10	3.98	8.12	
comcast	$ 0.14	0.76	5.38	10.57	
higgs	$ 1.68	30.20	27.13	-	
upsell	$ 0.21	0.23	5.40	9.96	
Linear Regression	**$ 0.41**	**12.05**	**1.26**	**3.22**	
glm-100k-500	$ 0.02	0.08	0.08	0.03	
glm-100m-10	$ 0.23	0.54	0.43	0.52	
glm-10k-2k	$ 0.01	0.11	0.15	0.08	
glm-1m-50	$ 0.02	0.05	0.08	0.05	
glm-50k-10k	$ 0.13	11.28	0.52	2.54	
Logistic Regression	**$ 0.76**	**24.47**	**1.36**	**4.97**	
log-100k-1k	$ 0.04	0.38	0.16	0.07	
log-100m-10	$ 0.39	0.61	0.40	0.62	
log-1m-50	$ 0.02	0.05	0.07	0.04	
log-500k-500	$ 0.07	0.51	0.20	0.17	
log-50k-10k	$ 0.25	22.92	0.51	4.06	
Random Forest	**$ 4.57**	**10.30**	**9.65**	**36.26**	
adult	$ 0.11	0.22	4.11	7.06	
amazon	$ 0.10	0.29	2.99	8.17	
comcast	$ 0.50	0.84	1.59	19.80	
higgs	$ 3.76	8.76	-	-	
upsell	$ 0.11	0.18	0.95	1.24	
Total Cost	**$ 7.85**	**78.21**	**57.77**	**81.90**	
SAS % Lower	**-**	**-90%**	**-86%**	**-90%**	

Appendix Table 2:

Cost to run tests, by library and data set

Examining the right side of Figure 1 for "Less Cost" the calculations are derived from the Appendix Table 2 above. As shown, the total cost for each platform is calculated as the sum of the runtime costs for completing all the tests. These are taken directly from the runtime per test, and the Azure instance size. It is important to note that when a test case failed to complete, no costs were attributed even though a user would incur runtime costs up until the point of failure. This provides a cost advantage to libraries that fail.

The "Percent Less" claim of 86% or greater is derived from the calculation of the three general purpose libraries, with results showing more than 86% reduction in costs. Moreover, generally the claim that "SAS-Viya is 86%+ Lower Cost" can be credibly made.

Data Sets

The datasets used for both training and scoring are available for download at the following URL.

- **SAS-Viya Performance Test Datasets (https://futurumgroup.com/sas-viya/)**

Shown below is the meta-data, i.e. the data about the data sets that were used for training and scoring the ML models tested.

Linear Regression	Train. Obsv.	Test Obs.	Features	Size (MB)
glm-100m-10	100,000,000	N/A	10	24,125
glm-1m-50	1,000,000	N/A	50	956
glm-100k-500	100,000	N/A	500	905
glm-10k-2k	10,000	N/A	2,000	360
glm-50k-10k	50,000	N/A	10,000	9,002

Appendix Table 3: Linear Regression Data Sets

Logistic Regression	Train. Obsv.	Test Obs.	Features	Size (MB)
log-100m-10	100,000,000	N/A	10	18,529
log-1m-50	1,000,000	N/A	50	918
log-500k-500	500,000	N/A	500	4,583
log-100k-1k	100,000	N/A	1,000	1,833
log-50k-10k	50,000	N/A	10,000	9,165

Appendix Table 4: Logistic Regression Data Sets

Random Forest	Train. Obsv.	Test Obs.	Features	Size (MB, Train)	Size (MB, Test)
adult	19,537	16,281	45	2.01	1.68
amazon	26,215	6,554	8	1.57	0.40
higgs	10,500,000	500,000	28	3,400.00	162.00
comcast	300,000	200,000	11	11.70	7.81
upsell	40,000	10,000	78	14.70	3.67

Appendix Table 5: Random Forest Data Sets

Gradient Boosting	Train. Obsv.	Test Obs.	Features	Size (MB, Train)	Size (MB, Test)
adult	19,537	16,281	45	2.01	1.68
amazon	26,215	6,554	8	1.57	0.40
higgs	10,500,000	500,000	28	3,400.00	162.00
comcast	300,000	200,000	11	11.70	7.81
upsell	40,000	10,000	78	14.70	3.67

Appendix Table 6: Gradient Boosting Data Sets

Appendix B – Test Results

In this section we present all test results for the 95 different configurations tested, including 30 for Gradient Boosting, 25 for Random Forest, and 20 each for linear regression and logistic regression totaling 95 different configurations. Each library was tested using 5 instance sizes, 8, 16, 32, 64 and 72 Azure CPUs for the specific instance types listed previously in Appendix A.

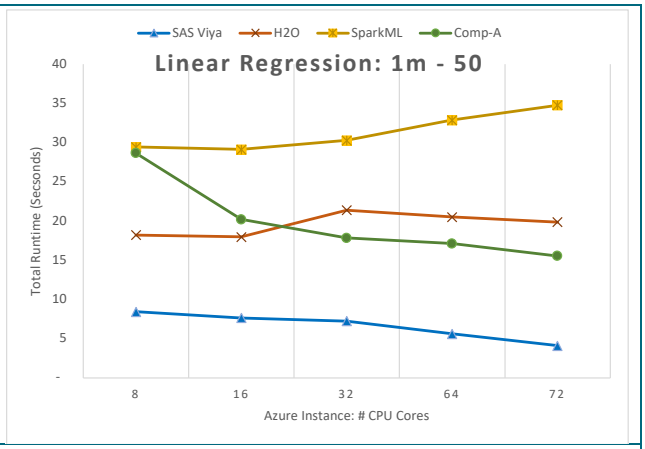
Linear Regression

In the following section we present the performance test results in both tabular format and in an accompanying graph to its right. For Linear Regression, all graphs are shown using a linear vertical axis, unlike the two preceding sections. The ML model type was "Ordinary Linear Regression", as regularization terms were not utilized.

Note 1: In the majority of cases, the smallest size instance (8 v CPUs) of SAS Viya outperformed all instances sizes of the competing general-purpose libraries (H2O, SparkML and Competitor-A).

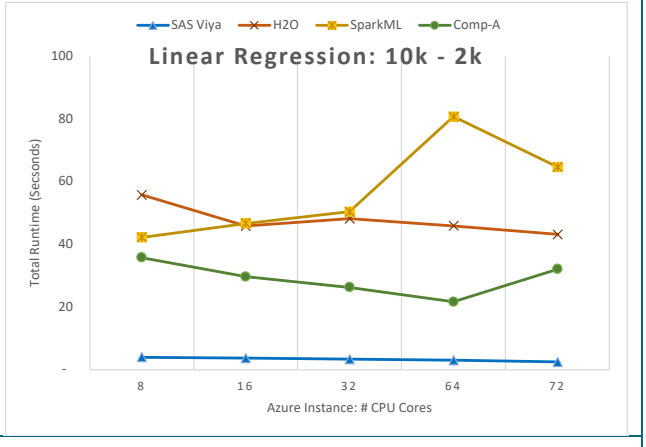
Note 2: SparkML and Competitor-A failed several or all test cases with a dataset and generally showed minor improvement for larger instances, while SAS Viya's runtimes decreased.

SAS Viya	H2O	SparkML	Comp-A
8.43	18.23	29.45	28.68
7.65	17.99	29.13	20.20
7.25	21.39	30.29	17.85
5.64	20.53	32.88	17.12
4.12	19.88	34.78	15.54



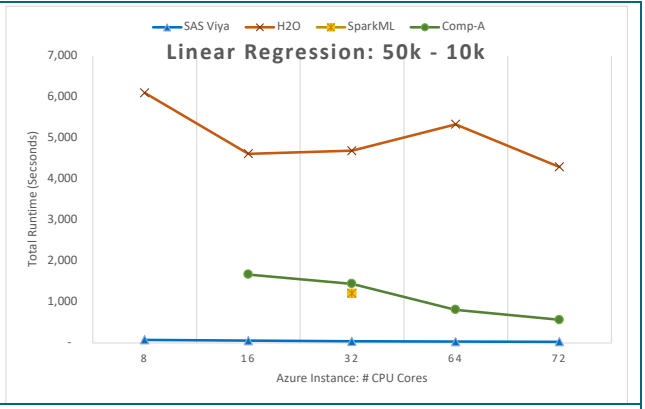
GLM 1m-50

SAS Viya	H2O	SparkML	Comp-A
4.03	55.85	42.26	35.81
3.75	45.89	46.73	29.70
3.39	48.25	50.45	26.31
3.11	45.93	80.83	21.69
2.50	43.24	64.73	32.11



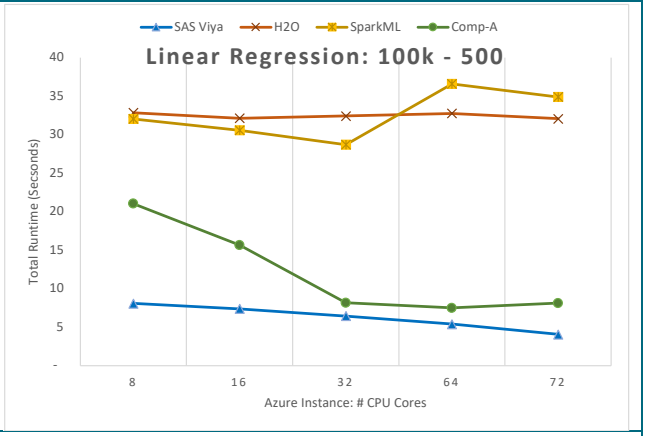
GLM 10k-2k

SAS Viya	H2O	SparkML	Comp-A
76.77	6,097.59		
56.48	4,613.43		1,672.44
43.92	4,687.91	1,215.51	1,443.61
35.82	5,333.28		811.90
31.21	4,293.71		567.18



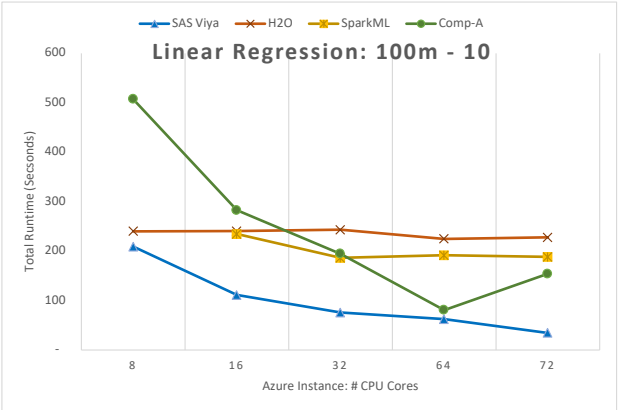
GLM 50k-10k

SAS Viya	H2O	SparkML	Comp-A
8.08	32.85	32.06	21.03
7.39	32.14	30.58	15.67
6.45	32.43	28.70	8.16
5.41	32.77	36.60	7.50
4.07	32.07	34.91	8.11



GLM 100k-500

SAS Viya	H2O	SparkML	Comp-A
209.12	240.16		507.16
111.51	240.75	234.62	282.74
75.91	243.36	186.51	194.59
62.46	224.84	191.54	81.02
34.99	227.74	188.39	154.25



GLM 100m-10

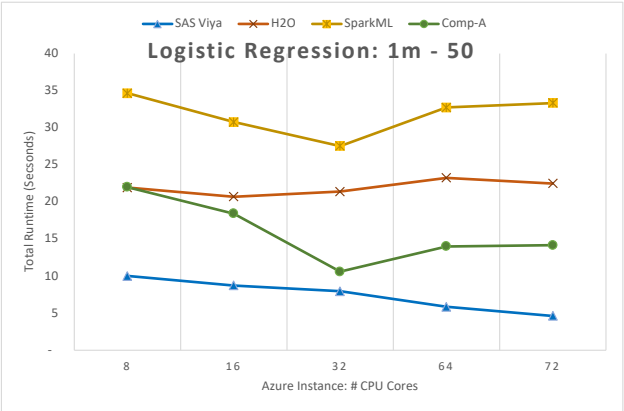
Logistic Regression

In the following section we present the performance test results in both tabular format and in an accompanying graph to its right. For Logistic Regression, all graphs are shown using a linear vertical axis, unlike the two preceding sections. The ML model type was "Ordinary Logistic Regression", as regularization terms were not utilized.

Note 1: In the majority of cases, the smallest size instance (8 v CPUs) of SAS Viya outperformed all instances sizes of the competing general-purpose libraries (H2O, SparkML and Competitor-A).

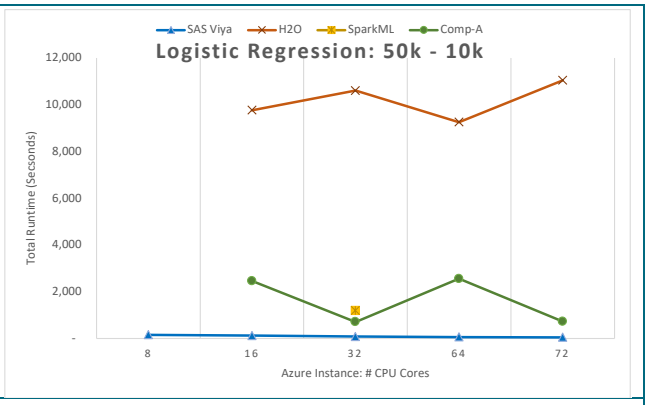
Note 2: SparkML failed several test cases and all competitors generally showed minor improvement for larger instances, while SAS Viya's runtimes decreased.

SAS Viya	H2O	SparkML	Comp-A
10.01	21.92	34.64	21.99
8.74	20.69	30.75	18.42
7.95	21.39	27.53	10.60
5.85	23.25	32.72	14.00
4.65	22.48	33.33	14.16



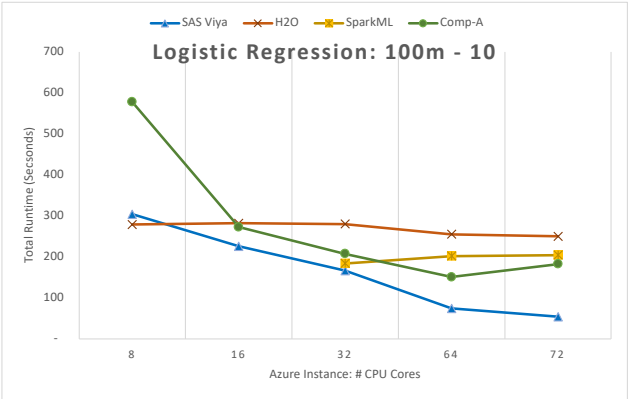
LOG 1m-50

SAS Viya	H2O	SparkML	Comp-A
160.90			
122.84	9,771.55		2,465.14
85.61	10,610.74	1,205.36	711.84
65.49	9,261.11		2,557.64
51.61	11,047.26		735.07



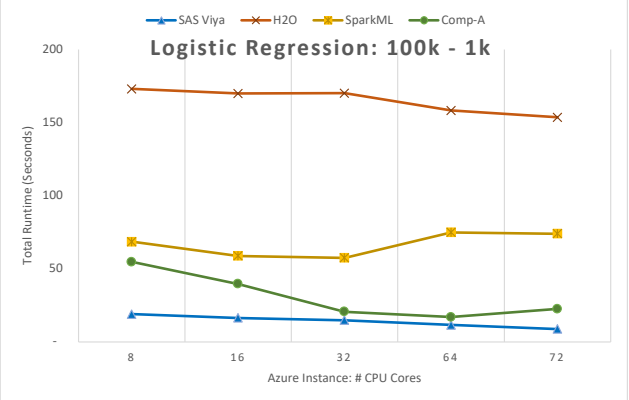
LOG 50k-10k

SAS Viya	H2O	SparkML	Comp-A
303.94	279.05		577.74
226.05	282.32		272.68
166.75	280.27	184.09	207.56
74.76	255.13	202.26	151.08
54.22	250.18	204.73	182.61



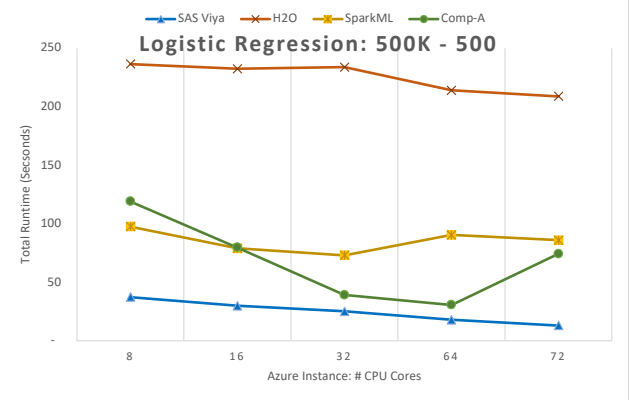
LOG 100m-10

SAS Viya	H2O	SparkML	Comp-A
19.09	173.04	68.54	54.74
16.49	169.90	58.89	39.73
14.91	170.02	57.55	20.72
11.76	158.23	75.00	17.07
8.98	153.58	74.03	22.70



LOG 100k-1k

SAS Viya	H2O	SparkML	Comp-A
37.31	236.45	97.61	119.13
29.95	232.42	79.00	79.73
25.24	233.84	73.13	39.17
17.90	214.00	90.55	30.65
13.07	208.79	85.97	74.30



LOG 500k-500

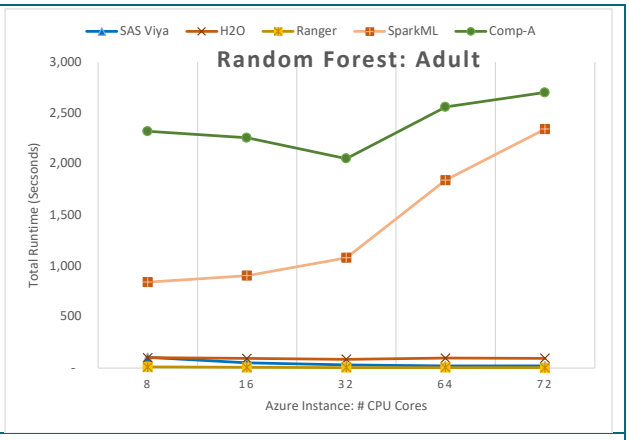
Random Forest

In the following section we present the performance test results in both tabular format and in an accompanying graph to its right. For Random Forest, all graphs are shown using a Lognormal vertical axis, due to the extreme differences between the best and worst performing libraries.

Note 1: In the majority of test cases, the smallest size instance (8 v CPUs) of SAS Viya outperformed the largest instance of all general-purpose libraries (H2O, SparkML and Competitor-A).

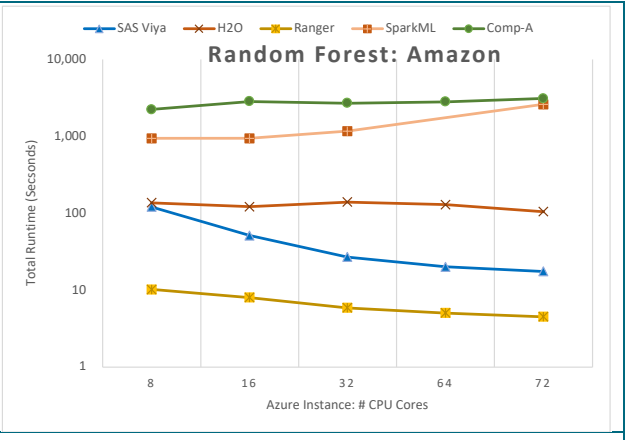
Note 2: Several competing libraries failed several or all test cases with a dataset and generally did not improve substantially for larger instances, while SAS Viya's runtimes decreased.

SAS Viya	H2O	Ranger	SparkML	Comp-A
104.63	101.31	10.04	842.69	2,320.20
51.64	94.16	7.72	906.10	2,256.28
31.15	87.09	5.79	1,082.08	2,054.19
22.58	98.90	4.83	1,844.18	2,559.51
21.22	96.73	4.00	2,342.67	2,701.84



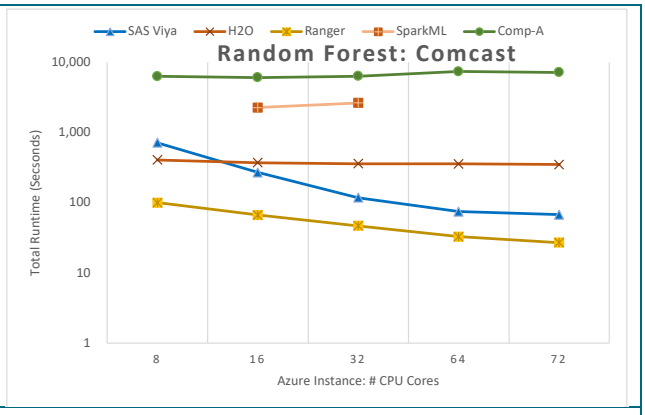
RF Adult

SAS Viya	H2O	Ranger	SparkML	Comp-A
120.80	136.80	10.18	939.58	2,229.08
51.02	121.11	7.98	939.55	2,834.00
26.94	139.43	5.89	1,167.25	2,693.20
20.09	129.44	5.03		2,802.60
17.52	104.80	4.50	2,596.19	3,093.64



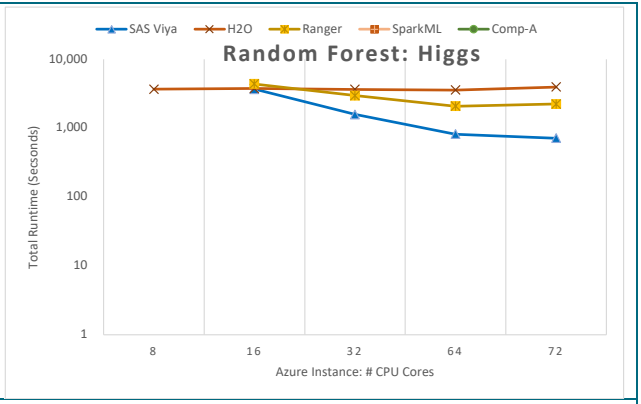
RF Amazon

SAS Viya	H2O	Ranger	SparkML	Comp-A
715.86	406.58	100.12		6,290.12
270.14	372.04	66.86	2,267.80	6,059.57
118.20	358.63	46.83	2,633.38	6,328.66
75.06	357.80	32.81		7,374.75
67.53	349.93	27.06		7,176.47



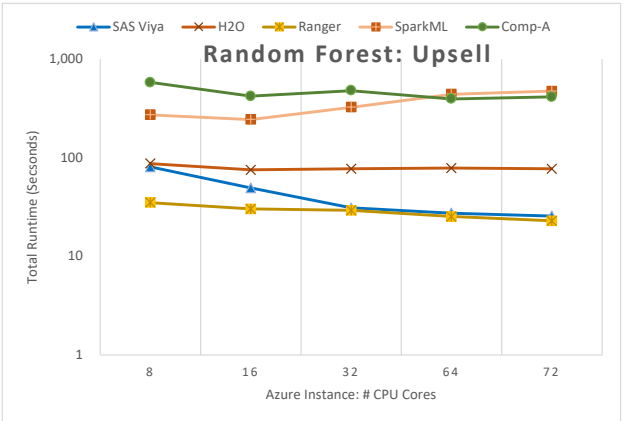
RF Comcast

SAS Viya	H2O	Ranger	SparkML	Comp-A
	3,657.00			
3,662.94	3,743.80	4,358.96		
1,576.69	3,643.83	2,953.99		
815.58	3,569.43	2,077.78		
714.88	3,944.46	2,240.18		



RF Higgs

SAS Viya	H2O	Ranger	SparkML	Comp-A
80.74	87.33	35.14	272.70	581.72
49.35	75.52	30.29	244.58	421.31
31.07	77.25	29.31	326.19	480.61
27.51	78.82	25.38	441.64	396.12
25.78	77.42	22.94	474.15	415.14



RF Upsell

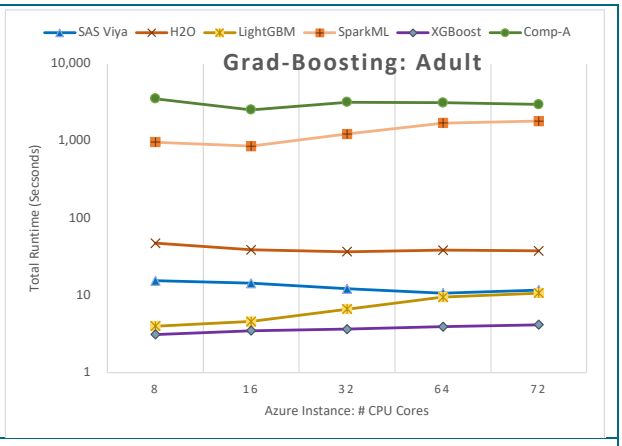
Gradient Boosting

In the following section we present the performance test results in both tabular format and in an accompanying graph to its right. For Gradient Boosting, all graphs are shown using a Lognormal vertical axis, due to the extreme differences between the best and worst performing libraries.

Note 1: For all 5 Gradient Boosting test cases, the smallest size instance (8 v CPUs) of SAS Viya outperformed all instance of all general-purpose libraries (H2O, SparkML and Competitor-A).

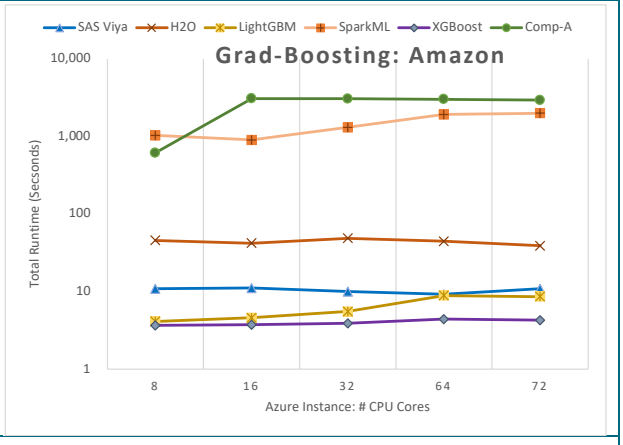
Note 2: SparkML and Competitor-A failed several configurations, in particular with the largest dataset named Higgs. Failures are shown in light red with no values, in order to enable proper charting.

SAS Viya	H2O	LightGBM	SparkML	XGBoost	Comp-A
15.44	47.51	4.01	963.78	3.12	3,523.68
14.39	38.88	4.59	855.29	3.49	2,528.18
12.20	36.72	6.63	1,230.27	3.66	3,169.76
10.71	38.60	9.47	1,705.43	3.94	3,126.34
11.76	37.83	10.69	1,805.99	4.17	2,967.61



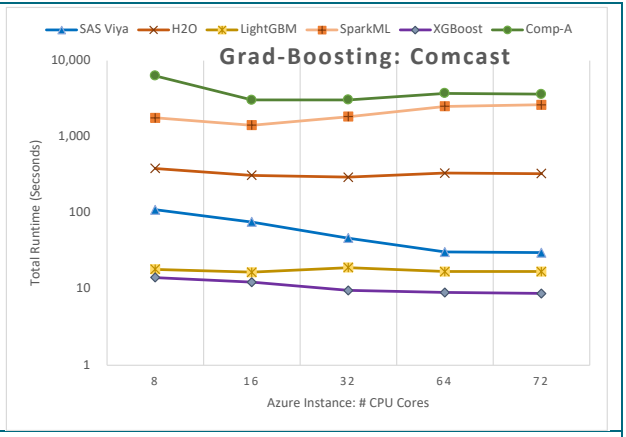
GB Adult

SAS Viya	H2O	LightGBM	SparkML	XGBoost	Comp-A
10.86	45.63	4.12	1,028.20	3.65	615.35
11.10	42.06	4.59	903.11	3.73	3,073.24
10.02	48.54	5.56	1,304.50	3.91	3,056.90
9.23	44.64	8.86	1,916.69	4.42	3,013.00
10.85	38.96	8.61	1,991.68	4.28	2,940.82



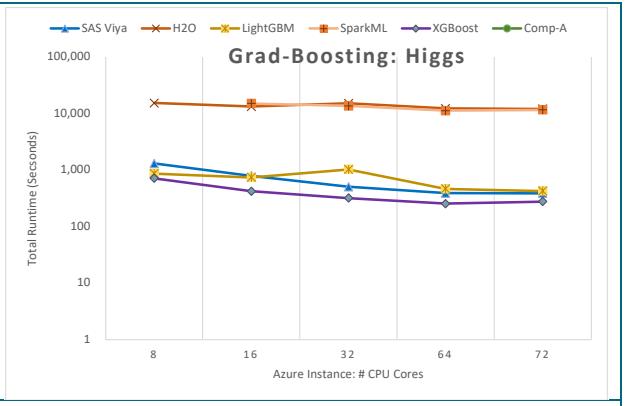
GB Amazon

SAS Viya	H2O	LightGBM	SparkML	XGBoost	Comp-A
109.90	382.83	18.17	1,766.32	14.20	6,289.79
76.03	311.26	16.66	1,411.81	12.33	3,031.68
46.71	293.71	19.11	1,832.25	9.59	3,045.74
30.64	333.52	17.00	2,503.20	9.05	3,707.55
30.09	327.93	16.95	2,627.35	8.75	3,624.40



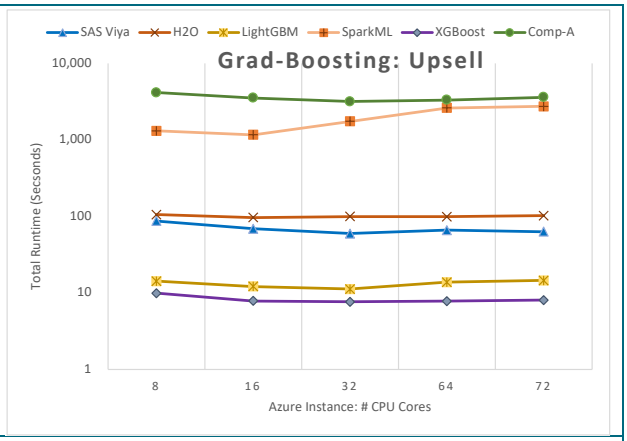
GB Comcast

SAS Viya	H2O	LightGBM	SparkML	XGBoost	Comp-A
1,296.16	15,292.19	863.31		712.67	
788.69	13,165.14	740.52	15,000.70	419.84	
503.44	15,033.24	1,021.43	13,544.62	317.74	
388.97	12,210.99	462.37	11,153.90	253.21	
387.36	12,025.43	423.98	11,548.71	275.26	



GB Higgs

SAS Viya	H2O	LightGBM	SparkML	XGBoost	Comp-A
86.76	105.11	14.26	1,311.93	9.90	4,133.78
68.70	96.61	12.11	1,159.38	7.83	3,518.46
59.65	99.32	11.21	1,739.97	7.64	3,157.88
66.09	98.73	13.79	2,602.22	7.82	3,314.80
62.99	101.92	14.50	2,730.53	8.05	3,601.91



GB Upsell

Additional Resources

The training and test datasets used for performance testing, along with the latest version of this paper and an Executive Summary are all available at the following location:

Futurum Group - SAS Viya Performance Resources: https://futurumgroup.com/sas-viya

- Executive Summary: Faster AI & Analytics: SAS Viya Outperforms the Competition
- Lab Insight: Comparing AI/ML Performance of SAS Viya vs. Alternatives
- Testing Datasets: Data used for performance testing including training and scoring

About The Futurum Group

The Futurum Group is dedicated to helping **IT professionals** and vendors create and implement strategies that make the most value of their storage and digital information. The Futurum Group services deliver **in-depth, unbiased analysis** on storage architectures, infrastructures, and management for IT professionals. Since 1997 The Futurum Group has provided services for thousands of end-users and vendor professionals through product and market evaluations, competitive analysis, and **education**.

This document was developed with funding from SAS Institute Inc. Although the document may utilize publicly available material from various vendors, including SAS and others, it does not necessarily reflect such vendors' positions on the issues addressed in this document.