*Please answer the following questions after examining these features:*

* *What do you think the most useful DASK feature is?*
* *Why is the advent of DASK so important?*
* *About what would you like to learn more?*

*Answer :*

*User Interfaces*

*Dask supports several user interfaces:*

* *High-Level*
  + [*Arrays*](https://docs.dask.org/en/stable/array.html)*: Parallel NumPy*
  + [*Bags*](https://docs.dask.org/en/stable/bag.html)*: Parallel lists*
  + [*DataFrames*](https://docs.dask.org/en/stable/dataframe.html)*: Parallel Pandas*
  + [*Machine Learning*](https://ml.dask.org)*: Parallel Scikit-Learn*
  + *Others from external projects, like [XArray](https://xarray.pydata.org)*
* *Low-Level*
  + [*Delayed*](https://docs.dask.org/en/stable/delayed.html)*: Parallel function evaluation*
  + [*Futures*](https://docs.dask.org/en/stable/futures.html)*: Real-time parallel function evaluation*

*Each of these user interfaces employs the same underlying parallel computing machinery, and so has the same scaling, diagnostics, resilience, and so on, but each provides a different set of parallel algorithms and programming style.*

*Dask is simply the most revolutionary tool for data processing. While Pandas and Numpy if were sometimes struggling with data that would not fit into RAM then Dask is definitely what is needed. Dask supports the Pandas dataframe and Numpy array data structures and is able to either be run on the local computer or be scaled up to run on a cluster. Essentially you write code once and then choose to either run it locally or deploy to a multi-node cluster using a just normal Pythonic syntax. This is a great feature in itself. The magic Dask feature, has been that with minimal code changes, we can run code code in parallel taking advantage of the processing power already on our laptop. Processing data in parallel, means less time to execute, less time to wait and more time to analyse!*

*Why Dask?*

* [*Python’s role in Data Science*](https://docs.dask.org/en/stable/why.html#python-s-role-in-data-science)

*This is fueled both by computational libraries like Numpy, Pandas, and Scikit-Learn and by a wealth of libraries for visualization, interactive notebooks, collaboration, and so forth.*

*Dask was developed to scale these packages and the surrounding ecosystem. It works with the existing Python ecosystem to scale it to multi-core machines and distributed clusters.*

* [*Dask has a Familiar API*](https://docs.dask.org/en/stable/why.html#dask-has-a-familiar-api)

*Dask provides ways to scale Pandas, Scikit-Learn, and Numpy workflows more natively, with minimal rewriting. It integrates well with these tools so that it copies most of their API and uses their data structures internally. Moreover, Dask is co-developed with these libraries to ensure that they evolve consistently, minimizing friction when transitioning from a local laptop, to a multi-core workstation, and then to a distributed cluster. Analysts familiar with Pandas/Scikit-Learn/Numpy will be immediately familiar with their Dask equivalents, and have much of their intuition carry over to a scalable context.*

* [*Dask Scales out to Clusters*](https://docs.dask.org/en/stable/why.html#dask-scales-out-to-clusters)

*DASK figures out how to break up large computations and route parts of them efficiently onto distributed hardware. Dask is routinely run on thousand-machine clusters to process hundreds of terabytes of data efficiently within secure environments.*

*Dask has utilities and documentation on how to deploy in-house, on the cloud, or on HPC super-computers. It supports encryption and authentication using TLS/SSL certificates. It is resilient and can handle the failure of worker nodes gracefully and is elastic, and so can take advantage of new nodes added on-the-fly. Dask includes several user APIs that are used and smoothed over by thousands of researchers across the globe working in different domains.*

* *[Dask Scales Down to Single Computers](https://docs.dask.org/en/stable/why.html" \l "dask-scales-down-to-single-computers)*

*Dask can empower analysts to manipulate 100GB+ datasets on their laptop or 1TB+ datasets on a workstation without bothering with the cluster at all. This can be preferable for the following reasons:*

1. *They can use their local software environment, rather than being constrained by what is available on the cluster or having to manage Docker images.*
2. *They can more easily work while in transit, at a coffee shop, or at home away from the corporate network*
3. *Debugging errors and analyzing performance is simpler and more pleasant on a single machine*
4. *Their iteration cycles can be faster*
5. *Their computations may be more efficient because all of the data is local and doesn’t need to flow through the network or between separate processes*

* [*Dask Integrates Natively with Python Code*](https://docs.dask.org/en/stable/why.html#dask-integrates-natively-with-python-code)

*Python includes computational libraries like Numpy, Pandas, and Scikit-Learn, and many others for data access, plotting, statistics, image and signal processing, and more. These libraries work together seamlessly to produce a cohesive*ecosystem*of packages that co-evolve to meet the needs of analysts in most domains today.*

*This ecosystem is tied together by common standards and protocols to which everyone adheres, which allows these packages to benefit each other in surprising and delightful ways.*

*Dask evolved from within this ecosystem. It abides by these standards and protocols and actively engages in community efforts to push forward new ones. This enables the rest of the ecosystem to benefit from parallel and distributed computing with minimal coordination. Dask does not seek to disrupt or displace the existing ecosystem, but rather to complement and benefit it from within.*

* [*Dask Supports Complex Applications*](https://docs.dask.org/en/stable/why.html#dask-supports-complex-applications)

*Dask helps to resolve these situations by exposing low-level APIs to its internal task scheduler which is capable of executing very advanced computations. This gives engineers within the institution the ability to build their own parallel computing system using the same engine that powers Dask’s arrays, DataFrames, and machine learning algorithms, but now with the institution’s own custom logic. This allows engineers to keep complex business logic in-house while still relying on Dask to handle network communication, load balancing, resilience, diagnostics, etc..*

* [*Dask Delivers Responsive Feedback*](https://docs.dask.org/en/stable/why.html#dask-delivers-responsive-feedback)

*Dask keeps users informed and content with a suite of helpful diagnostic and investigative tools including the following:*

1. *A*[*real-time and responsive dashboard*](https://docs.dask.org/en/stable/understanding-performance.html)*that shows current progress, communication costs, memory use, and more, updated every 100ms*
2. *A statistical profiler installed on every worker that polls each thread every 10ms to determine which lines in your code are taking up the most time across your entire computation*
3. *An embedded IPython kernel in every worker and the scheduler, allowing users to directly investigate the state of their computation with a pop-up terminal*
4. *The ability to reraise errors locally, so that they can use the traditional debugging tools to which they are accustomed, even when the error happens remotely.*

*I would like to learn more about Dask – ML*

# *Dask-ML*

*Dask-ML provides scalable machine learning in Python using [Dask](https://dask.org/) alongside popular machine learning libraries like*[*Scikit-Learn*](http://scikit-learn.org/)*, [XGBoost](https://ml.dask.org/xgboost.html), and others.*