ActiveState

MANAGEMENT'S GUIDE

UNLOCKING THE POWER OF DATA SCIENCE & MACHINE LEARNING WITH PYTHON

UNLOCKING THE POWER OF DATA SCIENCE & MACHINE LEARNING WITH PYTHON

ActiveState

INTRODUCTION: THE BIG DATA DILEMMA

In recent years, the amount of data available to companies has skyrocketed. According to IBM, 2.5 billion gigabytes (GB) of data are created every day.[1] With this massive influx comes new opportunities for companies to deliver greater customer experiences and get an edge on their competition. To this end, enterprises have been investing in big data platforms such as Hadoop, Spark and NoSQL databases. The greater challenge, though, is not only collecting and storing data, but being able to derive meaningful insights from it, and operationalizing those insights to create business value.

Data science and machine learning have emerged as the keys to unlocking this value. Unlike traditional business analytics, which focus on known values and past performance, data science aims to identify hidden patterns in order to drive new innovations. One of its main attributes is analyzing unstructured data, such as speech, images and text, as well as streaming data—such as sensor data and online behavior—which can be processed and acted upon in real time. From there, machine learning takes data mining to the next level by complementing human decisions with the ability to take actions automatically after detecting patterns.

Behind these efforts are the programming languages used by data science teams to clean up and prepare data, write and test algorithms, build statistical models, and translate into consumable applications or visualizations. In this regard, Python stands out as the language best suited for all areas of the data science and machine learning framework.

Designed as a flexible general purpose language, Python is widely used by programmers and easily learnt by statisticians. Its extensive libraries make it a powerful tool for statistical analysis, and it is routinely used to integrate models into web applications and production databases. Beyond conventional data analysis, Python is the leading language for machine learning, changing how businesses are operating in every industry.

This guide provides a summary of Python's attributes in each of these areas, as well as considerations for implementing Python to drive new insights and innovation from big data.

PYTHON VS. OTHER LANGUAGES

When it comes to which language is best for data science, the short answer is that it depends on the work you are trying to do. Python and R are suited for data science functions, while Java is the standard choice for integrating data science code into large-scale systems. However, Python challenges Java in that respect, and offers additional value as a tool for building web applications. Recently, Go has emerged as an up and coming alternative to the three major languages, but is not yet as well supported as Python.

In practice, data science teams use a combination of languages to play to the strengths of each one, with Python and R used in varying degrees. Below is a brief comparison table highlighting each language in the context of data science.

[1] Matthew Wall, "Big Data: Are you ready for blast-off?", BBC News: http://www.bbc.com/news/business-26383058

	Python	R	Java	Go
Core Strengths	Easy to use, multi-purpose, large community	Made for statistics, large community	High-performance, wide enterprise adoption	Modern architecture, clean reliable code, lightweight, fast
Ideal Use Cases	- Data analysis - Visualization - Exploratory analysis - Data engineering - Rapid prototyping - Machine learning - Web applications - Workflow integration	- Data analysis - Visualization - Exploratory analysis	Production systems	- Production systems - Data analysis, data engineering, machine learning (emerging) - Web applications - Microservices
Types of Users	- Data scientists - Data engineers - Machine learning engineers - Web app developers	- Data scientists - Data engineers	Systems developers	- Data scientists, data engineers, machine learning engineers (emerging) - Web app developers - Systems developers
Learning Curve	Accessible, easy to learn	Easy for statisticians. Steep, can be learned by programmers	Difficult, only for professional programmers	Steep, can be learned by programmers
Data Science Ecosystem	Robust, growing (PyPI)	Robust, mature (CRAN)	Lacks coverage	Early, growing
Performance	Reasonable, fast when using optimized libraries such as Intel	Slow	Fast	Fast, concurrent
Deployment	Many deployment tools/ integrations	Difficult, requires compiler	Simple; Java Virtual Machine (JVM) ubiquity	Simple (compiled executable)

Python

Python offers a strong combination of R's data analysis capabilities with general purpose speed and scalability. Data scientists who have a basic understanding of code can use Python effectively for day-to-day analyses, while developers can use Python to migrate code off of data scientists' machines into applications or production systems.

Although the Python ecosystem is still catching up to R, Python goes beyond R in areas such as machine learning and natural language processing. R's functionality can also be accessed within Python using the rpy2 library, giving users the best of both worlds. Python also supports exploratory analytics via Jupyter Notebook (formerly IPython Notebook)–a web application for sharing, explaining and iterating code.

From a deployment standpoint, architectural approaches such as containerization and microservices help ease the complexities of Python's various dependencies. As companies move towards microservices, they can use Python for the data-driven components of the system while existing components written in Java or other languages continue to operate uninterrupted.

R

Developed by and for statisticians, R is best suited for exploratory data analysis and visualization. Statisticians can use R to express their thoughts and ideas naturally without having a programming background. R is supported by a large and active community, and the CRAN repository contains thousands of packages and readily usable tests to perform almost any type of data analysis.

Because R is designed mainly for standalone computing, however, it is slower than Python and other languages, and is limited to working with datasets small enough to fit into memory. It also has a steep learning curve for those who are not trained statisticians. Data scientists will often use R for desktop prototyping and then use a more flexible language like Python or Java to deploy to production.

Java

Known for its performance and scalability, Java is often the preferred choice for enterprise infrastructure. It has a vast ecosystem and developer base, owing to its widespread enterprise adoption. As a compiled language, it is generally faster than interpreted languages, but for data science tasks, it is often slower than Python where exclusive libraries optimize Python's performance.

Compared to Python and R, Java is the least suited for statistical analysis and visualization. Although there are packages to add some of these functions, they are not as supported as those available for Python or R. Java's highly object-oriented language structure also make it extremely difficult to learn for non-professional programmers.

Scala, which runs on the JVM, is increasingly being used for machine learning and building high-level algorithms. However, like Java, it is not easily accessible to most data scientists due to its programmer-focused structure and lack of supporting libraries for data analysis.

Go

Recently, Go has emerged as an alternative to Python and R as a solution to issues around deployment and maintenance of data science code in production. This is because Go promotes more efficient, better quality, error-free code that is easily integrated into a company's existing architecture.[2] Popular packages like Jupyter Notebooks have also now been extended to support Go.

The main drawback to using Go for data science right now is that its ecosystem is underdeveloped compared to Python and R, missing essential tools for arrays and visualization. As a relatively new language, Go is quickly gaining traction for microservices and web applications. Keep your eye on this language for its potential productivity gains in data science.

DATA ANALYSIS WITH PYTHON

Aside from its flexibility and ease of use, Python's extensive libraries make it a powerful tool for data preparation, analysis and visualization compared to other languages. Along with foundational packages NumPy (for multi-dimensional arrays), SciPy (library of numerical algorithms) and Matplotlib (plotting and visualization), the following libraries give Python enhanced productivity and integration with big data sources.

[2] Daniel Whitenack, "Data Science Gophers", O'Reilly: https://www.oreilly.com/ideas/data-science-gophers

Complementing Your R Workforce With Python: rpy2

Python and R are often used together for their complementary capabilities. Data scientists may use R for its statistical functions and then wrap their model in a Python application that has a variety of additional features. Or they may use Python for analysis and call specialized packages only found in R. rpy2 provides an interface to access R within Python and is helpful for these use cases.

For those familiar with R, they can use rpy2 to learn some Python while thinking about their problem in R terms, and then express it in Python very easily. Those who are not familiar with either can use rpy2 to learn about R and Python at the same time, gaining the power of both languages.[3]

However, since rpy2 is calling the R libraries underneath, it is limited to working with data that fits into desktop or server memory. Analyzing large sets of data requires the use of frameworks such as Hadoop and HDF5, which are able to bring in data as you need it and push it out as you don't.

Connecting to Big Data Platforms

Python has all the libraries to connect to the various types of databases most organizations now use. This includes traditional SQL databases (i.e. mySQL, Microsoft SQL, Oracle), NoSQL databases (i.e. MongoDB, Cassandra, Redis), file systems (i.e. Hadoop), and streaming data (i.e. Kafka).

Aside from stored data, one of the big challenges companies face is how to deal with huge amounts of streaming data coming in from sources such as sensors, web feeds and market transactions. Analysis of streaming data can take a few forms. On one hand, companies can process streams of data and store them on disk for analysis and reporting as needed. Alternatively, companies may need to process and respond to data in real time (since that is when the data is most valuable). Examples of this include monitoring for service outages, making website recommendations and doing real-time price calculations.

For these purposes, Python connects to platforms that handle real-time data feeds, such as Kafka. Kafka has a variety of use cases for managing high-volume activities. For example, LinkedIn uses Kafka for activity stream data and operational metrics, while Netflix employs it for real-time monitoring and event processing.[4] Kafka is also a key component in Kubernetes, since log aggregation is crucial when deploying thousands of application instances at scale.

Python's ability to integrate and pull data from disparate sources and formats, both static and streamed, makes it extremely valuable as a means of generating return on investment (ROI) on an organization's big data infrastructure.

[3] Tom Radcliffe, "R vs. Python: A False Dichotomy", ActiveState Blog: https://www.activestate.com/blog/2016/02/r-vs-python-false-dichotomy
[4] Kafka website: https://kafka.apache.org/powered-by

Data Access	Data Prep	Algorithm and Model Building	Data Visualization	Operationalize Models
Hadoop	Pandas (data prep)	Fundamentals	Matplotlib (visualization, plotting)	Build into web app / (micro) services / APIs
SQL Server	Luigi (batch jobs)	NumPy	scikit-image (image processing)	Integrate into cloud services (AWS, Google)
mySQL	Dask (parallel computing)	SciPy	Bokeh (visualization in modern browsers)	Deploy on production systems
MongoDB	Airflow (batch, monitoring)	SymPy	Seaborn (statistical visualization)	
Redis		scikit-learn		
HDF5		Machine/Deep Learning		
		TensorFlow		
		Theano		
		Keras		
		Lasagne		
		NLKT (natural language)		
		Intel MKL (speed optimization)		

Figure 1. Sample of Python's tools and libraries for the data science workflow.

Preparing Messy, Missing and Unlabelled Data

Before any meaningful data analysis can take place, data must be cleaned up and organized into a useable format. Data preparation often involves labelling, filling in missing values and filtering outliers. Although it is essential to ensure accurate and reliable analysis, data preparation is considered a time consuming process, accounting for up to 80% of the work of data scientists. Python has a number of packages that facilitate the process of data preparation, helping data scientists focus more time on high value work.

One of these packages is Pandas, which brings the fundamental concept of data frames to Python. Data frames, previously unique to the R language, carry along entity labels as rows and headers in a matrix format. This allows data scientists to focus on analysis work while the data frames automate the "bookkeeping" of the metadata.[5] In addition, Pandas provides features such as missing data estimation, adding and deleting columns and rows, and handling time series. Pandas is useful for those who who prefer to work in Python, but prefer the R syntax, and offers the advantage of being able to call out to large datasets.

When it comes to working with data from various sources, packages such as Luigi, Airflow and Dask help with building data pipelines, managing workflow and

[5] Tom Radcliffe, "Pandas: Framing the Data", ActiveState Blog: https://www.activestate.com/blog/2017/05/pandas-framing-data

scaling up and scaling out analytics. These tools handle much of the complexity that arises as data science teams grow, move faster and build on an evolving data infrastructure, allowing them to keep their momentum on projects.

More Minds are Better than One: Jupyter (Formerly IPython) Notebook

Part of what makes Python great for data science is that it supports exploratory and interactive programming. Data scientists can easily share their code and annotations with colleagues, as well as experiment with code and see the results as they go along.

To start with, IPython adds an interactive command shell to Python which provides a number of development enhancements. One of these enhancements is Jupyter Notebook, a browser-based tool for authoring documents (notebooks) which combine code with explanatory text, mathematics, computations, diagrams and other media. Data scientists can use notebooks to keep their analysis and observations in one place and share them with colleagues or the community.

Notebooks are useful as a way to troubleshoot problems, since they allow others to easily see the steps one went through to try to solve them. They have also become popular as a medium for knowledge sharing. Many notebooks for data science are published and maintained on Github.

The other aspect of Jupyter Notebook is that it provides a browser-based REPL (Read-Eval-Print Loop). This allows users to enter an expression and evaluate the results immediately. For example, a data scientist can enter "import Pandas", create variables and get an output without having to compile or run the code. The availability of REPL within Python allows for instantaneous mathematical calculations, algorithm exploration and fast prototyping.

MACHINE LEARNING WITH PYTHON

The rise of big data has led to significant advances in artificial intelligence. Machine learning—the practice of using algorithms to train programs to not only recognize patterns in data, but to learn and take action when exposed to new data—has existed for many years as an approach to AI. Recently, though, thanks to the convergence of practically infinite data, storage, processing power and GPUs, we are now able to feed massive amounts of data through a system to train it. This has enabled the development of complex, multi-layered learning systems, called neural networks, creating the field of "deep learning".

Deep learning has led to the growth of a number of AI capabilities, including image recognition (i.e. recognizing objects or patterns in photographs) and natural language processing (i.e. summarizing text or recognizing speech). Its applications are far-reaching, from automatic stock trading and customer sentiment analysis to autonomous vehicles, optimized equipment usage, and new discoveries in healthcare, pharmaceuticals and other sciences.

Deep learning initiatives are primarily driven by open source languages. Machine learning workflows typically involve pre-processing to clean up the data, learning stages in which libraries of data are fed through a system to hone its pattern recognition, and testing of the results on independent data—followed by deployment if the tests are successful.

Python, in particular, is the most popular language for machine and deep learning. Python packages like Pandas and the Natural Learning Toolkit (NLTK) help with the pre-processing. TensorFlow, Theano and Keras, as well as scikit-learn, provide the algorithms, additional libraries, computational power and user-friendly control to develop the learning stages, and deployment is simply a matter of packaging the model once it's running well in testing.

Here is a look at some of the major Python packages in the machine learning workflow.

Processing and Analyzing Human Language: NLTK

Critical to the discussion of machine and deep learning is the ability to analyze unstructured data, such as natural language. While understanding language makes up the majority of human activities, the ability to process human language into meaningful information is an incredibly difficult task for machines. Nuances such as context, slang, misspellings and phrasing do not easily fit into a pre-defined database format. With the help of machine learning, natural language processing (NLP) is able to break language down into digestible units.

NLP applications are everywhere. Common examples are voice recognition software, search auto-completions and customer service chatbots. One important NLP function is summarization, the ability to summarize text such as news articles or research papers into executive briefs. Another example is sentiment analysis. By aggregating and summarizing social media posts, organizations can gage customer sentiment of their products or services, as well as indicators of what is driving positive or negative sentiment.

One of the most widely used NLP libraries is Python's Natural Language Toolkit (NLTK). NLTK provides essential functions such as tokenization (extracting key words and phrases from content), stemming (i.e. grouping words like happy, happiness and happier) and creating parse trees, which are tree diagrams that reveal linguistic structure and word dependencies. NLTK provides over 50 corpora[6]—repositories of names, words and phrases—as well as numerous algorithms and cookbooks, which would all be exceedingly difficult to build in-house.

Training AI Engines, Google Style: TensorFlow

Created by Google, TensorFlow is the most popular foundational library for building deep learning models. It provides the framework to train complex neural networks and is the AI engine that powers many of Google's operations, such as search ranking, image classification, drug discovery and NLP services like language translation and voice recognition. It is also open source, allowing enterprises to leverage the power created by machine learning researchers at Google to build their own neural networks with their own data.

Since its introduction two years ago, TensorFlow has been a sensation. It was the "most forked" project on GitHub in 2015[7] and currently has been forked over 28,000 times and has over 60,000 stars. TensorFlow is also closely aligned with Python, having been built on Python and C++. Although its API is available for C++, Haskell, Java and Go, it is primarily intended for Python as a bridge to the underlying C++ engine.

[6] NLTK Documentation: http://www.nltk.org/
[7] James Vincent, "Google has given its open-source machine learning software a big upgrade", The Verge: https://www.theverge.com/2016/4/13/11420144/google-machine-learning-tensorflow-upgrade

TensorFlow users include companies such as Airbnb, Airbus, Snapchat and Qualcomm[8]. One notable use case is UK online supermarket Ocado, which uses TensorFlow to route robots around its warehouses, improve demand forecasting and recommend items to add to customers' online shopping carts[9]. Ocado built their system in six months using Python, C++ and Kubernetes with TensorFlow—a project initiated by weather storms highlighting the need to prioritize emails in their contact centers based on their content rather than on a first-come, first-serve basis.

Enabling Faster, Large-Scale Computational Processing

Machine and deep learning require a great deal of processing power. Theano is a Python library that enables fast numerical computation by optimizing mathematical expressions. It is effectively a compiler that takes your structures and turns them into highly efficient code using multi-array (NumPy-like) syntax, native C code and a host of other optimizations, getting as much performance as possible out of CPUs and GPUs.

Theano is written in Python and has a Python interface. It was developed in 2007 at the University of Montreal, and was one the first libraries of its kind, and is considered an industry standard for the advancement of deep learning.

The Intel® Machine Kernel Library (MKL) provides another set of optimization libraries for math processing and neural network routines. Developed for science, engineering and financial applications, Intel MKL provides multi-threaded and vectorized functions that maximize performance of NumPy, SciPy, Theano and other computational libraries when running on Intel or compatible multi-core processors. Estimates range from two to ten times speedups on individual workstations, and much higher as more cores are applied to the model[10]. Since Intel MKL utilizes C and Fortran, it is compatible with many existing linear algebra libraries and offers Python performance comparable to C or C++.

Going from Idea to Result Faster with Keras

Keras is a user-friendly layer that can be used over top of either TensorFlow or Theano, both of which can be a little low-level for many deep learning use cases. It provides an easier way to build deep learning models, and is designed with the belief, "Being able to go from idea to result with the least possible delay is key to doing good research[11]."

While Theano and TensorFlow give fine-grained control over their underlying learning engines, Keras makes it easy to rapidly explore ideas. Keras might not be what you would use in production, but for exploratory deep learning it makes it extremely easy to get started quickly and do a rapid evaluation of different approaches to a problem. Just because Keras is user-friendly does not mean it's simple, though: it supports advanced machine learning algorithms like recurrent neural networks and noise layers.

[8] TensorFlow website: https://www.tensorflow.org/
[9] ComputerWorldUK, "What is TensorFlow, and how are businesses using it?" http://www.computerworlduk.com/open-source/what-is-tensorflow-how-are-businesses-using-it-3658374/
[10] Performance benchmarks available on Intel MKL website: https://software.intel.com/en-us/mkl/features/benchmarks
[11] Keras Documentation: https://keras.io/

RECOMMENDATIONS

The combination of flexibility and extensive libraries make Python the ideal language for data science and machine learning. So how do you get started with Python for your data science initiatives? You can download the default Python implementation (CPython), install the core packages for numerical and scientific computing—NumPy, SciPy and Matplotlib—and start exploring. Or, to make life easier, you can try an alternative Python implementation such as ActivePython with these libraries and many more pre-packaged.

In either case, implementing Python beyond a few internal machines or on production systems brings up a number of considerations. These include which Python distribution to standardize on, setup and configuration time, staffing, support and security requirements. Each of these factors depends on your organization's specific needs.

Here is a look at these key considerations.

Open Source vs. Commercial Tools

One of Python's great strengths is its open source ecosystem. Python libraries are constantly emerging and advancing based on the contributions of the community, which drives more innovation than can be provided by any one company. While some commercial Python-based platforms offer easy-to-use collaboration or visualization tools, customers can find themselves locked into a vendor-specific toolset. In contrast, the open source Python universe gives data scientists the flexibility to grab the right tool for the job at any time and run with it.

One example of open source innovation is TensorFlow, which has exploded in community contributions since its release as an open source library by Google in 2015. PyTables offers a separate example of open source synergy. Released in 2003 as a way to manage large amounts of data, PyTables has grown in tandem with HDF5, which started much later as part of the big data boom. Both examples demonstrate the value of open source in evolving with constantly changing business needs.

Staffing

Data scientists are notoriously hard to find and expensive to hire. As part of a multi-pronged approach to building data science teams, companies are retooling and training data scientists from within the organization. Implementing Python provides benefits for training and recruitment, such as employee growth opportunities, faster results from ramping up data science efforts in conjunction with training, and easier hiring of additional staff.

Data analysts familiar with R can learn Python with relative ease due to its low learning curve and frameworks like rpy2. Since Python can connect to all the data sources organizations use, data scientists-in-training can start mining big data for insights while learning on the job.

In addition, aligning with the open source Python ecosystem allows organizations to recruit skilled staff from a larger candidate pool. By bringing in people who are already experienced with Python, organizations can benefit from faster onboarding and consequently, faster time to market.

Getting Started

Although open source Python offers a wide selection of tools and libraries, setting up individual user environments can take a significant amount of time and resources. High-value staff can end up wasting days on the low-value work of installing and configuring packages before they are able to start writing algorithms.

To solve this challenge, specialized Python distributions come precompiled with the most popular open source packages for data science, including the SciPy stack and machine learning libraries. By using a precompiled distribution, data science and application development teams can stay focused on productivity, rather than having to hack together and maintain all the components they need.

Technical Support

Solving technical issues for open source Python implementations is a challenge. Aside from troubleshooting issues internally, organizations must resort to posting issues on public forums such as Stack Overflow, which can take days or weeks to get a response, if they get one at all. This can be impractical for time-sensitive or critical issues where downtime is not an option.

On top of that, many organizations are hesitant to reveal their intellectual property in a public forum, where questions on specific algorithms or machine learning packages could easily expose competitive advantages.

Based on these factors, commercial support could be a worthwhile or necessary investment.

Licensing and Security

Licence compliance risks are surprisingly common in commercial applications. According to a recent Black Duck report, up to 85% of audited code bases were found to be out of compliance with open source license terms[12], exposing organizations to potentially costly legal challenges. To address this problem, certain commercial Python providers offer full license reviews of the packages included in their distributions, as well as legal indemnification to protect against potential IP infringement lawsuits arising from the use of third-party software.

Often times, open source components are added directly to code bases with security vulnerabilities present. According to Black Duck, more than 60% of audited applications contained open-source vulnerabilities. With hundreds of open source packages in various ecosystems, and organizations' lack of oversight over these components, it is easy for data engineers or data scientists to accidentally download vulnerabilities, unbeknownst to their IT departments.

Commercial Python distributions can provide greater security, since the packages are generally reviewed and maintained by the commercial provider. When using a precompiled distribution, you can check with the provider to ensure that all included packages are vetted for security vulnerabilities, that the latest secure versions of packages are included, and that all packages are monitored for security updates on an ongoing basis.

[12] Madison Moore, SD Times, "Black Duck audit highlights risk of open-source security vulnerabilities"
http://sdtimes.com/black-duck-audit-highlights-risk-open-source-security-vulnerabilities/
[13] Gartner, "Gartner Says It's Not Just About Big Data; It's What You Do With It: Welcome to the Algorithmic Economy"
http://www.gartner.com/newsroom/id/3142917

CONCLUSION: BECOMING AN ALGORITHMIC BUSINESS

As companies continue to invest in big data, the issue is becoming less about the data itself, but rather how the data is used to create competitive products and services. According to Gartner, "Companies will be valued not just on their big data, but on the algorithms that turn that data into actions and impact customers[13]".

Python is the fundamental tool for this purpose, serving as a common language for the multi-disciplinary field of data science. It allows data scientists to interrogate data from disparate sources, developers to turn those insights into applications, and systems engineers to deploy on any infrastructure, whether on-premise or in the cloud. With Python, companies are able to get the most ROI out of their existing investments in big data.

Companies are not only maximizing their use of data, but transforming into "algorithmic businesses" with Python as the leading language for machine learning. Whether it's automatic stock trading, discoveries of new drug treatments, optimized resource production or any number of applications involving speech, text or image recognition, machine and deep learning are becoming the primary competitive advantage in every industry.

The time is now for companies to get started on data science initiatives if they have not already. Introducing Python into their technology stack is an important step, but companies should consider factors such as support requirements, staffing plans, licensing compliance and security. By addressing these needs early on, data science teams can focus on unlocking the power of their data and driving innovation forward.

ABOUT ACTIVEPYTHON

ActivePython is a leading Python distribution used by large enterprises, government and community developers. With over 300 of the top open source packages included for data science, machine learning, web application and general Python development, ActivePython delivers proven open source software with enterprise-level security and support. ActivePython is made by ActiveState, a founding member of the Python Software Foundation, trusted by millions of developers and 97% of Fortune-1000 companies.

Getting started with ActivePython for data science is easy. Your team can start writing algorithms for free with Community Edition, and learn more about commercial options for use in production at www.activestate.com

ActiveState Software Inc.
business-solutions@activestate.com

Phone: **+1.778.786.1100**
Fax: **+1.778.786.1133**

Toll-free in North America:
1.866.631.4581

ActiveState®

ABOUT ACTIVESTATE

ActiveState, the Open Source Languages Company, believes that enterprises gain a competitive advantage when they are able to quickly create, deploy, and efficiently manage software solutions that immediately create business value, but they face many challenges that prevent them from doing so. The Company is uniquely positioned to help address these challenges through our experience with enterprises, people and technology. ActiveState is proven for the enterprise: More than two million developers and 97% of Fortune-1000 companies use ActiveState's end-to-end solutions to develop, distribute, and manage their software applications. Global customers like Bank of America, CA, Cisco, HP, Lockheed Martin and Siemens trust ActiveState to save time, save money, minimize risk, ensure compliance, and reduce time to market.