In this lesson, you have learned:

* How Big Data is defined by the Vs: Velocity, Volume, Variety, Veracity, and Value.
* How Hadoop and other tools, combined with distributed computing power,  are used to handle the demands of Big Data.
* What skills are required to analyse Big Data.
* About the process of Data Mining, and how it produces results.

Hadoop :

Yahoo hired someone named Doug Cutting who had been working on a clone or a copy of the Google big data architecture and now that's called Hadoop. And if you google Hadoop you'll see that it's now a very popular term and there are many, many, many if you look at the big data ecology there are hundreds of thousands of companies out there that have some kind of footprint in the big data world.

At the bottom of data science you see probability and statistics. You see algebra, linear algebra you see programming and you see databases.now have the computational capabilities to apply some new techniques - machine learning. Where now we can take really large data sets and instead of taking a sample and trying to test some hypothesis we can take really, really large data sets and look for patterns. And so back off one level from hypothesis testing to finding patterns that maybe will generate hypotheses.

The term big data refers to data sets that are so massive, so quickly built, and so varied that they defy traditional analysis methods such as you might perform with a relational database. The concurrent development of enormous compute power in distributed networks and new tools and techniques for data analysis means that organizations now have the power to analyze these vast data sets. A new knowledge and insights are becoming available to everyone. There is no one definition of Big Data, but there are certain elements that are common across the different definitions, such as velocity, volume, variety, veracity, and value.

These are the V's of Big Data.

**Velocity** is the speed at which data accumulates. Data is being generated extremely fast, in a process that never stops. Near or real-time streaming, local, and cloud-based technologies can process information very quickly.Every 60 seconds, hours of footage are uploaded to YouTube which is generating data. Think about how quickly data accumulates over hours, days, and years.

**Volume** is the scale of the data, or the increase in the amount of data stored. Drivers of volume are the increase in data sources, higher resolution sensors, and scalable infrastructure. The world population is approximately seven billion people and the vast majority are now using digital devices; mobile phones, desktop and laptop computers, wearable devices, and so on. These devices all generate, capture, and store data -- approximately 2.5 quintillion bytes every day. That's the equivalent of 10 million Blu-ray DVD's.

**Variety** is the diversity of the data. Structured data fits neatly into rows and columns, in relational databases while unstructured data is not organized in a pre-defined way, like Tweets, blog posts, pictures, numbers, and video. Variety also reflects that data comes from different sources, machines, people, and processes, both internal and external to organizations. Drivers are mobile technologies, social media, wearable technologies, geo technologies, video, and many, many more. Let's think about the different types of data; text, pictures, film, sound, health data from wearable devices, and many different types of data from devices connected to the Internet of Things.

**Veracity** is the quality and origin of data, and its conformity to facts and accuracy. Attributes include consistency, completeness, integrity, and ambiguity. Drivers include cost and the need for traceability. With the large amount of data available, the debate rages on about the accuracy of data in the digital age. Is the information real, or is it false?

80% of data is considered to be unstructured and we must devise ways to produce reliable and accurate insights. The data must be categorized, analyzed, and visualized. Data Scientists today derive insights from Big Data and cope with the challenges that these massive data sets present. The scale of the data being collected means that it’s not feasible to use conventional data analysis tools.

However, alternative tools that leverage distributed computing power can overcome this problem. Tools such as Apache Spark, Hadoop and its ecosystem provide ways to extract, load, analyze, and process the data across distributed compute resources, providing new insights and knowledge. This gives organizations more ways to connect with their customers and enrich the services they offer.

So next time you strap on your smartwatch, unlock your smartphone, or track your workout, remember your data is starting a journey that might take it all the way around the world, through big data analysis, and back to you.

**Value** is our ability and need to turn data into value. Value isn't just profit. It may have medical or social benefits, as well as customer, employee, or personal satisfaction. The main reason that people invest time to understand Big Data is to derive value from it.

**Data mining** is the process of automatically searching and analyzing data, discovering previously unrevealed patterns. It involves preprocessing the data to prepare it and transforming it into an appropriate format. Once this is done, insights and patterns are mined and extracted using various tools and techniques ranging from simple data visualization tools to machine learning and statistical models.

**Machine learning** is a subset of AI that uses computer algorithms to analyze data and make intelligent decisions based on what it is learned without being explicitly programmed. Machine learning algorithms are trained with large sets of data and they learn from examples. They do not follow rules-based algorithms. Machine learning is what enables machines to solve problems on their own and make accurate predictions using the provided data.

**Deep learning** is a specialized subset of machine learning that uses layered neural networks to simulate human decision-making. Deep learning algorithms can label and categorize information and identify patterns. It is what enables AI systems to continuously learn on the job and improve the quality and accuracy of results by determining whether decisions were correct. Ex: speech recognition, people, images, classifying images…

**Artificial neural networks**, often referred to simply as neural networks, take inspiration from biological neural networks, although they work quite a bit differently. A neural network in AI is a collection of small computing units called neurons that take incoming data and learn to make decisions over time. Neural networks are often layer-deep and are the reason deep learning algorithms become more efficient as the data sets increase in volume, as opposed to other machine learning algorithms that may plateau as data increases. They were abandoned for some time because they were computationally very expensive. Neural Networks on steroids will get us Deep learning

Now that you have a broad understanding of the differences between some key AI concepts, there is one more **differentiation that is important to understand that between Artificial Intelligence and Data Science**. Data Science is the process and method for extracting knowledge and insights from large volumes of disparate data. It's an interdisciplinary field involving mathematics, statistical analysis, data visualization, machine learning, and more. It's what makes it possible for us to appropriate information, see patterns, find meaning from large volumes of data and use it to make decisions that drive business. Data Science can use many of the AI techniques to derive insight from data. For example, it could use machine learning algorithms and even deep learning models to extract meaning and draw inferences from data. There is some interaction between AI and Data Science, but one is not a subset of the other. Rather, Data Science is a broad term that encompasses the entire data processing methodology while AI includes everything that allows computers to learn how to solve problems and make intelligent decisions. Both AI and Data Science can involve the use of big data. That is, significantly large volumes of data.

Where do we apply machine learning:

1. recommender systems are certainly one of the major applications. Classifications, cluster analysis, trying to find some of the marketing questions. Ex: u followed this person u might want to follow this one, U might want to watch or buy something….
2. predictive analytics is another area of machine learning. We're using new techniques to predict things that statisticians don't particularly like. Decision trees, Bayesian Analysis, naive Bayes, lots of different techniques. Precision versus recall and the problems of over sampling and over fitting so you can, someone who knows a little about data science can apply these techniques but they really need to know, maybe not the details of the technique as much as how, what the trade-offs are.
3. Fraud detection

Regression is a statistical technique developed by Sir Frances Galton.

* Data Science helps physicians provide the best treatment for their patients, and helps meteorologists predict the extent of local weather events, and can even help predict natural disasters like earthquakes and tornadoes.
* That companies can start on their data science journey by capturing data. Once they have data, they can begin analysing it.
* Some ways that data is generated by consumers.
* How businesses like Netflix, Amazon, UPs, Google, and Apple use the data generated by their consumers and employees.
* The purpose of the final deliverable of a Data Science project is to communicate new information and insights from the data analysis to key decision-makers.
* Data Scientists need programming, mathematics, and database skills, many of which can be gained through self-learning.
* Companies recruiting for a Data Science team need to understand the variety of different roles Data Scientists can play, and look for soft skills like storytelling and relationship building as well as technical skills.
* High school students considering a career in Data Science should learn programming, math, databases, and, most importantly practice their skills.
* The length and content of the final report will vary depending on the needs of the project.
* The structure of the final report for a Data Science project should include a cover page, table of contents, executive summary, detailed contents, acknowledgements, references and appendices.
* The report should present a thorough analysis of the data and communicate the project findings.

The 10 components of a report are:

1. **Cover Page**:

Where we should mention the following:

* + 1. Title of Report
    2. Names of author
    3. Affiliations
    4. Contacts
    5. Name of institutional Publisher
    6. Date of Publication

1. **Table of Content**
2. **Abstract/Executive Summary**
3. **Introductory Section:** Here we need to explain the problem to the reader.
4. **Literature Review:** It follows the introductory section with reviews of relevant research on the subject matter, quotes and highlights on gaps in the existing knowledge which the analysis will fill. This is the part where we formally introduce the research question and hypothesis.
5. **Methodology Section:** Here we need to introduce the research Methods and Data Sources used for Analysis. In case the data is new we need to explain the Data collection exercise. At this stage we need to refer to the literature review to support our choices of variables, data and methods and how they will help us to answer our research question.
6. **Results Section:** Here we present the empirical findings using descriptive statistics, illustrative graphics for plots and spatial Data Analytics for maps. Then we formally put our hypothesis to test by running statistical models, regression Models, Categorial Analysis, Time series data, Mining Data…
7. **Discussion Section:** Here we craft our argument by building on the results mentioned earlier in the previous sections. In this section we rely heavily on the narrative to communicate the results to the reader by referring to the main question asked in this research and knowledge gaps identified and how our findings provide the ultimate missing pieces.
8. **Conclusion:** Here we generalize the findings and take a marketing approach to promote the findings. It is important at this stage to identify future developments in research and applications that could result from the research
9. **References – Acknowledgement - Appendices**

**Lesson 2 : Tools for Data Science**

Communities such as user!, WhyR?, SatRdays, and R-Ladies are all great to connect with.

**SQL – structured Query Language**

1. some people say “sequel.”
2. While the acronym stands for “Structured Query Language,” many people do not consider SQL to be like other software development languages because it's a non-procedural language and its scope is limited to querying and managing data.
3. While it is not a “data science” language per se, data scientists regularly use it because

it's simple and powerful!

1. it's much older than Python and R, by about 20 years, having first appeared in 1974.
2. SQL was developed at IBM!
3. This language is useful in handling structured data; that is, the data incorporating relations among entities and variables. A relational database is formed by collections of two-dimensional tables; for example, datasets and Microsoft Excel spreadsheets.
4. because it's so pervasive and easy to use, SQL interfaces for many NoSQL and big data repositories have also been developed.
5. The SQL language is subdivided into several language elements, including clauses, expressions, predicates, queries, and statements.
6. SQL will help you do many different jobs in data science, including business and

data analyst, and it's a must in data engineering.

1. with SQL, you access the data directly. There's no need to copy it. This can speed up workflow executions considerably.
2. SQL is the interpreter between you and the database. SQL is an American National Standards Institute, or "ANSI," standard, which means if you learn SQL and use it with one database, you will be able to easily apply that SQL knowledge to many other databases. There are many different SQL databases available, including MySQL, IBM Db2, PostgreSQL, Apache, OpenOffice Base, SQLite, Oracle, MariaDB, Microsoft SQL Server, and more.

Other Languges:

Scala, Java, C++, and Julia are probably the most traditional data science languages on this slide. But JavaScript, PHP, Go, Ruby, Visual Basic, and others have all found their place in the data science community as well!

**Java :** Java is a tried-and-true general-purpose object oriented programming language. It's been widely adopted in the enterprise space and is designed to be fast and scalable. Java applications are compiled to bytecode and run on the Java Virtual Machine, or "JVM." Some notable data science tools built with Java include Weka, for data mining; Java-ML, which is a machine learning library; Apache MLlib, which makes machine learning scalable; and Deeplearning4j, for deep learning, Apache Hadoop is another Java-built application.

It manages data processing and storage for big data applications running in clustered systems.

**Scala** is a general-purpose programming language that provides support for functional programming and a strong static type system. Many of the design decisions in the construction of the Scala language were made to address criticisms of Java.

Scala is also interoperable with Java, as it runs on the JVM.

The name "Scala" is a combination of "scalable" and "language." This language is designed to grow along with the demands of its users.

For data science, the most popular program built using Scala is Apache Spark. Spark is a fast and general-purpose cluster computing system. It provides APIs that make parallel jobs easy to write, and an optimized engine that supports general computation graphs.

Spark includes Shark, which is a query engine; MLlib, for machine learning; GraphX, for graph processing; and Spark Streaming. Apache Spark was designed to be faster than Hadoop.

**C++** is a general-purpose programming language. It is an extension of the C programming language, or "C with Classes.” C++ improves processing speed, enables system programming, and provides broader control over the software application.

Many organizations that use Python or other high-level languages for data analysis and exploratory tasks still rely on C++ to develop programs that feed that data to customers in real-time. For data science, a popular deep learning library for dataflow called TensorFlow was built with C++. But while C++ is the foundation of TensorFlow, it runs on a Python interface, so you don’t need to know C++ to use it. MongoDB, a NoSQL database for big data management, was built with C++. Caffe is a deep learning algorithm repository built with C++, with Python and MATLAB bindings.

A core technology for the World Wide Web, **JavaScript** is a general-purpose language that extended beyond the browser with the creation of Node.js and other server-side approaches.

Javascript is NOT related to the Java language.

For data science, the most popular implementation is undoubtedly TensorFlow.js. TensorFlow.js makes machine learning and deep learning possible in Node.js as well as in the browser. TensorFlow.js was also adopted by other open source libraries, including brain.js and machinelearn.js. The R-js project is another great implementation of JavaScript for data science. R-js has re-written linear algebra specifications from the R Language into Typescript.

This re-write will provide a foundation for other projects to implement more powerful math base frameworks like Numpy and SciPy of Python. Typescript is a superset of JavaScript.

Julia was designed at MIT for high-performance numerical analysis and computational science. It provides speedy development like Python or R, while producing programs that run as fast as C or Fortran programs.

Julia is compiled, which means that the code is executed directly on the processor as executable code; it calls C, Go, Java, MATLAB, R, Fortran, and Python libraries; and has refined parallelism. The Julia language is relatively new, having been written in 2012, but it has a lot of promise for future impact on the data science industry.

JuliaDB is a particularly useful application of Julia for data science. It's a package for working with large persistent data sets.

**Open Source tools for Data Science:**

1. **Data Management** is the process of persisting and retrieving data. The most widely used open source data management tools are relational databases such as MySQL and PostgreSQL; NoSQL databases such as MongoDB Apache CouchDB, and Apache Cassandra; and file-based tools such as the Hadoop File System or Cloud File systems like Ceph. Finally,Elasticsearch is mainly used for storing text data and creating a search index for fast document retrieval.
2. **Data Integration and Transformation**, often referred to as Extract, Transform, and Load, or “ETL,” is the process of retrieving data from remote data management systems. Transforming data and loading it into a local data management system is also part of Data Integration and Transformation.  These days, data scientists often propose the term “ELT” – Extract, Load, Transform“ELT”, stressing the fact that data is dumped somewhere and the data engineer or data scientist themself is responsible for data. Another term for this process has now emerged: “data refinery and cleansing.” Here are the most widely used open source data integration and transformation tools: Apache AirFlow, originally created by AirBNB; KubeFlow, which enables you to execute data science pipelines on top of Kubernetes; Apache Kafka, which originated from LinkedIn; Apache Nifi, which delivers a very nice visual editor; Apache SparkSQL (which enables you to use ANSI SQL and scales up to compute clusters of 1000s of nodes), and NodeRED, which also provides a visual editor. NodeRED consumes so little in resources that it even runs on small devices like a Raspberry Pi.
3. **Data Visualization** is part of an initial data exploration process, as well as being part of a final deliverable. A similar approach uses Hue, which can create visualizations from SQL queries. Kibana, a data exploration and visualization web application, is limited to Elasticsearch (the data provider). Finally, Apache Superset is a data exploration and visualization web application.
4. **Model Building** is the process of creating a machine learning or deep learning model using an appropriate algorithm with a lot of data.
5. **Model deployment** makes such a machine learning or deep learning model available to third-party

applications. Model deployment is extremely important. Once you’ve created a machine learning model capable of predicting some key aspects of the future, you should make that model consumable by other developers and turn it into an API. Apache PredictionIO currently only supports Apache Spark ML models for deployment, but support for all sorts of other libraries is on the roadmap. Seldon is an interesting product since it supports nearly every framework, including TensorFlow, Apache SparkML, R, and scikit-learn. Seldon can run on top of Kubernetes and Redhat OpenShift. Another way to deploy SparkML models is by using MLeap. Finally, TensorFlow can serve any of its models using the TensorFlow service. You can deploy to an embedded device like a Raspberry Pi or a smartphone using TensorFlow Lite, and even deploy to a web browser using TensorFlow dot JS.

1. **Model monitoring and assessment** ensures continuous performance quality checks on the deployed

models. These checks are for accuracy, fairness, and adversarial robustness. Once you’ve deployed a machine learning model, you need to keep track of its prediction performance as new data arrives in order to maintain outdated models. ModelDB is a machine model metadatabase where information about the models are stored and can be queried. It natively supports Apache Spark ML Pipelines and scikit-learn. A generic, multi-purpose tool called Prometheus is also widely used for machine learning model monitoring, although it’s not specifically made for this purpose. A generic, multi-purpose tool called Prometheus is also widely used for machine learning model monitoring, although it’s not specifically made for this purpose. Model performance is not exclusively measured through accuracy. Model bias against protected groups like gender or race is also important. The IBM AI Fairness 360 open source toolkit does exactly this. It detects and mitigates against bias in machine learning models. Machine learning models, especially neural-network-based deep learning models, can be subject to adversarial attacks, where an attacker tries to fool the model with manipulated data or by manipulating the model itself. The IBM Adversarial Robustness 360 Toolbox can be used to detect vulnerability to adversarial attacks and help make the model more robust. Machine learning modes are often considered to be a black box that applies some mysterious “magic.” The IBM AI Explainability 360 Toolkit makes the machine learning process more understandable by finding similar examples within a dataset that can be presented to a user for manual comparison. The IBM AI Explainability 360 Toolkit can also illustrate training for a simpler machine learning model by explaining how different input variables affect the final decision of the model.

1. **Code asset management** uses versioning and other collaborative features to facilitate teamwork. For code asset management – also referred to as version management or version control – Git is now the standard. Multiple services have emerged to support Git, with the most prominent being GitHub, which provides hosting for software development version management. The runner-up is definitely GitLab, which has the advantage of being a fully open source platform that you can host and manage yourself. Another choice is Bitbucket.
2. **Data asset management** brings the same versioning and collaborative components to data. Data asset management also supports replication, backup, and access right management. Data asset management, also known as data governance or data lineage, is another crucial part of enterprise grade data science. Data has to be versioned and annotated with metadata. Apache Atlas is a tool that supports this task. Another interesting project, ODPi Egeria, is managed through the Linux Foundation and is an open ecosystem. It offers a set of open APIs, types, and interchange protocols that metadata repositories use to share and exchange data. Finally, Kylo is an open source data lake management software platform that provides extensive support for a wide range of data asset management tasks.
3. **Development environments**, commonly known as Integrated Development Environments, or “IDEs”,

are tools that help the data scientist to implement, execute, test, and deploy their work. Jupyter first emerged as a tool for interactive Python programming; it now supports more than a hundred different programming languages through “kernels.” Kernels shouldn’t be confused with operating system kernels. Jupyter kernels are encapsulating the different interactive interpreters for the different programming languages. A key property of Jupyter Notebooks is the ability to unify documentation, code, output from the code, shell commands, and visualizations into a single document. JupyterLab is the next generation of Jupyter Notebooks and in the long term, will actually replace Jupyter Notebooks. The architectural changes being introduced in JupyterLab makes Jupyter more modern and modular. From a user’s perspective, the main difference introduced by JupyterLab is the ability to open different types of files, including Jupyter Notebooks, data, and terminals. You can then arrange these files on the canvas. Although Apache Zeppelin has been fully reimplemented, it’s inspired by Jupyter Notebooks and provides a similar experience. One key differentiator is the integrated plotting capability. In Jupyter Notebooks, you are required to use external libraries in Apache Zeppelin, and plotting doesn’t require coding. You can also extend these capabilities by using additional libraries. RStudio is one of the oldest development environments for statistics and data science, having been introduced in 2011. It exclusively runs R and all associated R libraries. However, Python development is possible and R is therefore tightly integrated into this tool to provide an optimal user experience. RStudio unifies programming, execution, debugging, remote data access, data exploration, and visualization into a single tool. Spyder tries to mimic the behaviour of RStudio to bring its functionality to the Python world. Although Spyder does not have the same level of functionality as RStudio, data scientists do consider it an alternative. But in the Python world, Jupyter is used more frequently.

1. **Execution environments** are tools where data preprocessing, model training, and deployment

take place. Sometimes your data doesn’t fit into a single computer’s storage or main memory capacity. That’s where cluster execution environments come in. The well known cluster-computing framework Apache Spark is among the most active Apache projects and is used across all industries, including in many Fortune 500 companies. The key property of Apache Spark is linear scalability. This means, if you double the number of servers in a cluster, you’ll also roughly double its performance. After Apache Spark began to gain market share, Apache Flink was created. The key difference between Apache Spark and Apache Flink is that Apache Spark is a batch data processing engine, capable of processing huge amounts of data file by file. Apache Flink, on the other hand, is a stream processing image, with its main focus on processing real-time data streams. Although engine supports both data processing paradigms, Apache Spark is usually the choice in most use cases. One of the latest developments in the data science execution environments is called “Ray,” which has a clear focus on large-scale deep learning model training. Let’s look at open source tools for data scientists that are fully integrated and visual. With these tools, no programming knowledge is necessary. Most important tasks are supported by these tools; these tasks include data integration, transformation, data visualization, and model building. KNIME originated at the University of Konstanz in 2004. As you can see, KNIME has a visual user interface with drag-and-drop capabilities. It also has built-in visualization capabilities. Knime can be be extended by programming in R and Python, and has connectors to Apache Spark. Another example of this group of tools is Orange. It’s less flexible than KNIME, but easier to use.

**Commercial Tools for Data Science**

1. **Data management:** most of an enterprise’s relevant data is stored in an Oracle Database, Microsoft SQL Server, or IBM Db2. Although open source databases are gaining popularity, those three data management products are still considered the industry-standard. They won’t disappear in the near future. It’s not just about functionality. Data is at the heart of every organization, and the availability of commercial supports plays a major role. Commercial supports are delivered directly from software vendors, influential partners, and support networks.
2. **Data integration**: we’re talking about “extract, transform, and load,” or “ETL” tools. According to a Gartner Magic Quadrant, Informatica Powercenter and IBM InfoSphere DataStage are the leaders, followed by products from SAP, Oracle, SAS, Talend, and Microsoft. These tools support design and deployment of ETL data-processing pipelines through a graphical interface. They also provide connectors to most of the commercial and open source target information systems. Finally, Watson Studio Desktop includes a component called Data Refinery, which enables the defining and execution of data integration processes in a spreadsheet style.
3. **Data visualizations** are utilizing business intelligence, or “BI”, tools. Their main focus is to create visually attractive and easy-to-understand reports and live dashboards. The most prominent commercial examples are: Tableau, Microsoft Power BI, and IBM Cognos Analytics. Another type of visualization targets data scientists rather than regular users. A sample problem might be “How can different columns in a table relate to each other?” This type of functionality is contained in Watson Studio Desktop. If you want to build a machine learning model using a commercial tool, you should consider using a data mining product. The most prominent of these types of products are: SPSS Modeler and SAS Enterprise Miner. In addition, A version of SPSS Modeler is also available in Watson Studio Desktop, based on the cloud version of the tool.
4. **Model deployment** is tightly integrated in the model building process. This diagram shows an example of the SPSS Collaboration and Deployment Services which are used to deploy any type of asset created by the SPSS software tools suite. Other vendors use the same type of process. Commercial software can also export models in an open format. For example, SPSS Modeler supports the exporting of models as Predictive Model Markup Language, or PMML, which can be read by many other commercial and open software packages.
5. **Model monitoring** is a new discipline and there are currently no relevant commercial tools available. As a result, open source is the first choice.
6. **Code asset management**.: is a new discipline and there are currently no relevant commercial tools available. As a result, open source is the first choice. Open source with Git and GitHub is the effective standard.
7. **Data asset management**, often called data governance or data lineage, is a crucial part of enterprise grade data science. Data must be versioned and annotated using metadata. Vendors, including Informatica Enterprise Data Governance and IBM, provide tools for these specific tasks. The IBM InfoSphere Information Governance Catalog covers functions like data dictionary, which facilitates discovery of data assets. Each data asset is assigned to a data steward -- the data owner. The data owner is responsible for that data asset and can be contacted. Data lineage is also covered; this enables a user to track back through the transformation steps followed in creating the data assets. The data lineage also includes a reference to the actual source data. Rules and policies can be added to reflect complex regulatory and business requirements for data privacy and retention. Watson Studio is a fully integrated development environment for data scientists. It’s usually consumed through the cloud, and we’ll cover more about it in a later lesson. There is also a desktop version available. Watson Studio Desktop combines Jupyter Notebooks with graphical tools to maximize data scientists’ performance. Watson Studio, together with Watson Open Scale, is a fully integrated tool covering the full data science life cycle and all the tasks we’ve discussed previously. just keep in mind that they can be deployed in a local data center on top of Kubernetes or RedHat OpenShift. Another example of a fully integrated commercial tool is H2O Driverless AI, which covers the complete data science life cycle.

Bolster = support

machine learning and deep learning models.

Data contains a wealth of information that can be used to solve certain types of problems. Traditional data analysis approaches, such as a person manually inspecting the data or a specialized computer program that automates the human analysis, quickly reach their limits due to the amount of data to be analyzed or the complexity of the problem. Machine learning uses algorithms – also known as ”models” - to identify patterns in the data. The process by which the model learns these patterns from data is called “model training." Once a model is trained, it can then be used to make predictions. When the model is presented with new data, it tries to make predictions or decisions based on the patterns it has learned from past data. Machine learning models can be divided into three basic classes: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning is one of the most commonly used type of machine learning models. In supervised learning, a human provides input data and the correct outputs. The model tries to identify relationships and dependencies between the input data and the correct output. Generally speaking, supervised learning is used to solve regression and classification problems. Let’s look at an example for each problem type: Regression models are used to predict a numeric, or “real," value. For example, given information about past home sales, such as geographic location, size, number of bedrooms, and sales price, you can train a model to predict the estimated sales price for other homes with similar characteristics. Classification models are used to predict whether something belongs to a category, or “class." For example, given a set of emails along with a designation of whether or not they are considered spam, an algorithm can be trained to identify unsolicited emails. In unsupervised learning, the data is not labelled by a human. The models must analyze the data and try to identify patterns and structure within the data based only on the characteristics of the data itself. Clustering and anomaly detection are two examples of this learning style. Clustering models are used to divide each record of a data set into one of a small number of similar groups. An example of a clustering model could be providing purchase recommendations for an e-commerce store based on past shopping behavior and the content of a shopping basket. Anomaly detection identifies outliers in a data set, such as fraudulent credit card transactions or suspicious online log-in attempts. The third type of learning, reinforcement learning, is loosely based on the way human beings and other organisms learn. Think about a mouse in a maze. If the mouse gets to the end of the maze it gets a piece of cheese. This is the “reward” for completing a task. The mouse learns – through trial and error – how to get through the maze to get as much cheese as it can. In a similar way, a reinforcement learning model learns the best set of actions to take, given its current environment, in order to get the most reward over time. This type of learning has recently been very successful in beating the best human players in games such as go, chess, and popular strategy video games. Deep learning is a specialized type of machine learning. It refers to a general set of models and techniques that tries to loosely emulate the way the human brain solves a wide range of problems. It is commonly used to analyze natural language, both spoken and text, as well as images, audio, and video, to forecast time series data and much more. Deep learning has had a lot of recent success in these and other areas and is therefore becoming an increasingly popular and important tool for data science. Deep learning typically requires very large data sets of labeled data to train a model, is compute-intensive, and usually requires special purpose hardware to achieve acceptable training times. You can build a custom deep learning model from scratch or use pre-trained models from public model repositories. Deep learning models are implemented using popular frameworks such as TensorFlow, PyTorch, and Keras. Deep learning frameworks typically provide a Python API, and many support other programming languages, such as C++ and JavaScript. You can download pre-trained state-of-the-art models from repositories that are commonly referred to as model "zoos." Popular model zoos include those provided by TensorFlow, PyTorch, Keras, and ONNX. Models are also published by academic and commercial research groups. While it is beyond the scope of this video to explain in detail how you would go about building a model, let’s briefly outline the high-level tasks using an example. Assume you want to enable an application to identify objects in images by training a deep learning model. First, you collect and prepare data that will be used to train a model. Data preparation can be a time-consuming and labor-intensive process. In order to train a model to detect objects in images, you need to label the raw training data by, for example, drawing bounding boxes around objects and labeling them. Next, you build a model from scratch or select an existing model that might be well suited for the task from a public or private resource. You then train the model on your prepared data. During training, your model learns from the labeled data how to identify objects that are depicted in an image. Once training has commenced, you analyze the training results and repeat the process until the trained model performance meets your requirements. When the trained model performs as desired, you deploy it to make it available to your applications.