**Tools for Data Science**

Welcome to the course!

You've begun one of the most complete overviews on data science tooling that you’ll currently

find on the internet.

This doesn’t mean that we cover each and every tool, but later in the course we’ll

introduce a comprehensive list of tasks a data scientist needs to perform and give you

the top two or three open source and commercial tools available to complete them.

We also explain how the tools overlap in functionality, what their pros and cons are, and how these

tools can address the whole data science pipeline.

Let’s start with data.

Data is obviously central to data scientists.

In this course, we’ll show you how to manage, extract, transform, analyze, and visualize

data.

Now, you might be able to survive data science without programming skills if you use the

right set of tools.

However, we highly recommend getting familiar with programming and the related programming

frameworks for data science.

To help you along, we’ll introduce you to the most commonly used programming languages

and frameworks available for data science.

That said, there is a lot of automation available in the latest tooling that a data scientist

can use.

In this course, we’ll explain how to make use of those tools to save time and uncover

inspiration.

Visual programming is available in many tools.

In this course, you’ll learn how visual programming can be used to speed up development

time and to help non-programmers enter the field of data science.

Open source software is leading the field of data science, but its total costs of ownership,

or "TCO," can be higher at times due to configuration, customization and maintenance costs.

As a result, commercial software also has its place, especially since the new generation

of commercial data science software leverages open source software and open standards.

This makes it easy to migrate between tools and can reduce overall TCO.

In this course, we’ll introduce you to both open source and commercial software and point

out their strengths and weaknesses for data science.

We'll also show you ways that you can take advantage of their respective strengths.

Finally, we'll show you how cloud computing can be used to further speed up and facilitate

data scientists' work.

We'll introduce you to the most commonly used and newly emerging cloud tools for data science.

In addition to lectures, this course, has numerous labs to make you more familiar with

the material and get hands-on experience.

There are also multiple quizzes to test your learning.

Nothing more to say than we’re glad to have you in the course and happy learning.

In case you have trouble in any way, please don’t hesitate to contact us in the discussion

forum.

There's nothing left but to begin!

We're very happy to have you with us as you start your data science journey.

If you have any trouble with any of the course material, please don’t hesitate to contact

us in the discussion forum.

Let's get started!

**Languages of Data Science**

The languages of Data Science

For anyone just getting started on their data science journey, the range of technical options

can be overwhelming. There is a dizzying amount of choice when it comes to programming languages.

Each has it's own strengths and weaknesses and there is no one right answer to the question

of which one you should learn first. The answer to that question depends largely on your needs,

the problems you are trying to solve, and who you are solving them for.

Python, R, and SQL are the languages that we recommend you consider first and foremost.

But there are so many others that have their own strengths and features.

Scala, Java, C++, and Julia are some of the most popular.

Javascript, PHP, Go, Ruby, and Visual Basic all have their own unique use cases as well.

The language you choose to learn will depend on the things you need to accomplish and the

problems you need to solve. It will also depend on what company you work for, what role you

have, and the age of your existing application. We’ll explore the answers to this question

as we dive into the popular languages in the data science industry.

There are many roles available for people who are interested in getting involved in

data science. Business Analyst

Database Engineer Data Analyst

Data Engineer Data Scientist

Research Scientist Software Engineer

Statistician Product Manager

Project Manager and many more.

Let’s dive into what we will learn in Lesson 1. We will put most of our focus on the top

three Data Science languages: Python, R, and SQL, which each have their own lessons. Then

we will go on to highlight other noteworthy languages and what makes them special. Then

we’ll finish with a short quiz to test your knowledge!

● 1.1.1 - Python ● 1.1.2 - R

● 1.1.3 - SQL ● 1.1.4 - Other Noteworthy Data Science

Languages ● 1.1.5 - Practice Quiz

**Introduction to Python**

1. Powerhouse language
   1. Most popular for data science
   2. Kaggle, Glassdoor: 75% used python
2. Why is it so popular:
   1. Less code
   2. 80% data professionals use Python
   3. Data science, AI, Machine Learning, web development and IoT devices like Raspberry Pi
   4. Global community shepherded by Python Software Foundation
3. High-level general-purpose language applied to many different problems
4. Large standard library providing tools to different tasks
   1. Databases, automation, web scraping, text processing, image processing, machine learning, data analytics
5. Data science Python specific computing libraries such as
   1. Pandas, NumPy, SciPy, Matpotlib
6. AI it has:
   1. TensorFlow, PyTorch, Keras and Scikit-learn
7. Can be used for Natural Language Processing (NLP) using Natural Language Toolkit (NLTK)
8. Python community, Pyladies

In this video, we will review the high-level features of the Python programming language.

Python is a powerhouse language.

It is by far the most popular programming language for data science.

According to the 2019 Kaggle Data Science and Machine Learning Survey, 75% of the over

10,000 respondents from around the world reported that they use Python on a regular basis.

Glassdoor reported that in 2019 more than 75% of data science positions listed included

Python in their job descriptions.

When asked which language an aspiring data scientist should learn first, most data scientists

say Python.

You are probably thinking, why on earth is Python so popular?

Well, let’s start with the people who use Python.

If you already know how to program, then Python is great for you because it uses clear, readable

syntax.

You can do many of the things you are used to doing in other programming languages but

with Python you can do it with less code.

If you want to learn to program, it’s also a great starter language because of the huge

global community and wealth of documentation.

In fact, several different surveys in 2019 found that over 80% of data professionals

worldwide use Python.

Python is useful for many situations, including data science, AI and machine learning, web

development, and IoT devices like the Raspberry Pi.

Large organizations that use Python heavily include IBM, Wikipedia, Google, Yahoo!, CERN,

NASA, Facebook, Amazon, Instagram, Spotify, and Reddit.

Python is a powerful general-purpose programming language that can do a lot of things.

It is widely supported by a global community and shepherded by the Python Software Foundation.

1.

Python is a high-level general-purpose programming language that can be applied to many different

classes of problems.

2.

It has a large, standard library that provides tools suited to many different tasks, including

but not limited to databases, automation, web scraping, text processing, image processing,

machine learning, and data analytics.

3.

For data science, you can use Python's scientific computing libraries such as Pandas, NumPy,

SciPy, and Matplotlib.

4.

For artificial intelligence, it has TensorFlow, PyTorch, Keras, and Scikit-learn.

5.

Python can also be used for Natural Language Processing (NLP) using the Natural Language

Toolkit (NLTK).

Another great selling point is the Python community, which has a well documented history

of paving the way for diversity and inclusion efforts in the tech industry as a whole.

The Python language has a code of conduct executed by the Python Software Foundation

that seeks to ensure safety and inclusion for all, in both online and in person python

communities.

There are also communities like PyLadies that seek to create spaces for people interested

in Python to learn in safe and inclusive environments.

PyLadies is an international mentorship group with a focus on helping more women become

active participants and leaders in the Python open source community.

**Introduction to R**

1. Kaggle: Learning 3 languages can increase salary greatly
2. Selling Points:
   1. Free to use, but a GNU (general public library) product (not open source but free software)
   2. Open Source: Business focused. Free Software: Set of values
   3. Allows for pubic collaboration and private and commercial use
   4. Widely supported globally
3. Statisticians, mathematicians and data miners for statistical software, graphing and data analysis
   1. Array-oriented syntax: easier to translate from math to code
   2. Largest repository of statistical knowledge
   3. 2018, R has released more than 15000 publicly released packages
   4. Integrates other computer languages: C++, Java, .Net and Python
   5. Matrix manipulation work straight out of the box
   6. Stronger oriented programming facilities than most statistical computing languages
   7. User!, WhyR?, SatRdays and R-ladies

In this video, we will give a brief overview of the R programming language.

After our last video on Python, where we discussed its wide adoption, you might be wondering

why on earth you should consider learning any other language.

Well, according to the results of the 2019 Kaggle Data Science survey, which had over

10k respondents from around the world, learning up to three languages can increase your salary!

And R has a lot to offer you.

Like Python, R is free to use, but it's a GNU project -- instead of being open source,

it's actually free software.

So if Python is open source and R is free software, what’s the difference?

Well, Both open source and free software commonly refer to the same set of licenses.

Many open source projects use the GNU General Public License, for example.

Both open source and free software support collaboration.

In many cases (but not all), these terms can be used interchangeably.

The Open Source Initiative (OSI) champions open source while the Free Software Foundation

(FSF) defines free software.

Open source is more business focused, while free software is more focused on a set of

values.

Back to why you should learn R. Because this is a free software project, you can use the

language in the same way that you contribute to open source, and it allows for public collaboration

and private and commercial use.

Plus, R is another language supported by a wide global community of people passionate

about making it possible to use the language to solve big problems.

Who is R for?

It's most often used by statisticians, mathematicians, and data miners for developing statistical

software, graphing, and data analysis.

The language’s array-oriented syntax makes it easier to translate from math to code,

especially for someone with no or minimal programming background.

According to Kaggle’s Data Science and Machine Learning Survey, most folks learn R when they're

a few years into their data science career, but it remains a welcoming language to those

who don’t have a software programming background.

R is popular in academia but companies that use R include IBM, Google, Facebook, Microsoft,

Bank of America, Ford, TechCrunch, Uber, and Trulia.

● R has become the world’s largest repository of statistical knowledge.

● As of 2018, R has more than 15,000 publicly released packages, making it possible to conduct

complex exploratory data analysis.

● R integrates well with other computer languages, such as C++, Java, C, .Net, and

Python.

● Common mathematical operations such as matrix multiplication work straight out of

the box.

● R has stronger object-oriented programming facilities than most statistical computing

languages.

There are many ways to connect with other R users around the globe.

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

And you can also check out the R project website for R conferences and events.

**Introduction to SQL**

1. Structured Query Language: Non-procedural language (scope is limited to querying and managing data
2. Not a data science language per se, data science use it because it is simple and powerful
3. 20 years older than Python, 1974, Developed at IBM.
4. Relational database formed by collection of two-dimensional tables
   1. E.g. datasets and Excel
5. SQL interfaces many NoSQL and big data repositories
6. Subdivided into several language elements
   1. Clauses, expressions, predications, queries and statements
7. Selling Points:
   1. Find many different jobs (business analyst and data analyst)
   2. Must in Engineering
   3. Access data directly, no need to copy beforehand – speed up workflow executions
   4. SQL interpreter between you and database
8. Many SQL databases available:
   1. MySQL, IBM Db2, PostgreSQL, Apache OpenOffice Base, SQLite, Oracle, MariaDB, Microsoft SQL Server, etc.

In this video, we'll take a high-level look at SQL.

SQL is a bit different from the other languages we’ve covered so far.

First off, it's formally pronounced “ess cue el,” although some people say “sequel.”

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.

While it is not a “data science” language per se, data scientists regularly use it because

it's simple and powerful!

Another couple of neat facts about SQL: it's much older than Python and R, by about 20

years, having first appeared in 1974.

And, SQL was developed at IBM!

This language is useful in handling structured data; that is, the data incorporating relations

among entities and variables.

SQL was designed for managing data in relational databases.

Here you can see a diagram showing the general structure of a relational database.

A relational database is formed by collections of two-dimensional tables; for example, datasets

and Microsoft Excel spreadsheets.

Each of these tables is then formed by a fixed number of columns and any number of rows.

BUT!

Even though SQL was originally developed for use with relational databases, because it's

so pervasive and easy to use, SQL interfaces for many NoSQL and big data repositories have

also been developed.

The SQL language is subdivided into several language elements, including clauses, expressions,

predicates, queries, and statements.

So what makes SQL great?

Knowing SQL will help you do many different jobs in data science, including business and

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

When performing operations with SQL, you access the data directly.

There's no need to copy it beforehand.

This can speed up workflow executions considerably.

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.

The syntax of the SQL you write might change a little bit based on the relational database

management system you’re using.

If you are looking to learn SQL you would be best served to focus on a specific relational

database and then plug into the community for that specific platform.

There are also many great introductory courses on SQL available!

**Other Languages**

1. Scala, Java, C++ and Julia most traditional
2. JavaScript, PHP, Go Ruby, Visual Basic also in data science
3. Java
   1. General purpose object oriented programming language
   2. Used in enterprise space, designed to be fast and scalable
   3. Java applications run on the Java Virtual Machine (JVM)
   4. Notable tools built with Java
      1. Weka: data mining
      2. Java-ML: Machine learning
      3. Apache MLlib: makes machine learning scalable
      4. Deeplearning4j: deep learning
      5. Apache Hadoop: Manages data processing and storage for big data running in clustered systems
      6. “Scala” (Scalable language)
      7. Apache Spark: Fast and general-purpose cluster computing system
         1. Provides API’s that make parallel jobs easy to write
         2. Optimized engine supports general computation graphs
      8. Shark: Query engine
      9. MLlib for machine learning
      10. GraphX for graph processing
      11. Spark Streaming
4. C++: Programs that feed data to clients in real-time
   1. TensorFlow: Built with C++ but runs on Python interface
   2. MongoDB: a NoSQL database for big data management
   3. Caffee: Deep learning algorithm repository with Python and MATLAB bindings
5. Java-Script: Not related to Java language
   1. TenserFlow.js: Makes machine learning and deep learning possible in Node.js as well as in browser
      1. Adopted by other source libraries: Brain.js and machinelearn.js
   2. R-js Project: Re-written linear algebra specifications from the R language into Typescript. Allows for powerful math based frameworks like Numpy and SciPy of Python.
   3. Typescript is a superset of Javascript
6. Julia (Designed MIT): High performance programs that run as cast as C or Fortran programs
   1. Compiled: Meaning the code is executed directly on the processor as executable code
   2. It calls C, Go Java, MATLAB, R, FORTRAN, and Python libraries and has refined parallelism.
   3. Written in 2012 – lots of promise for data science community
   4. JuliaDB: Useful for Data Science. Works with large persistent data sets

So far, we’ve reviewed Python, R, and SQL.

In this video, we will review some other languages that have compelling use cases for data science.

Ok, so indisputably, Python, R, and SQL are the three most popular languages that data

scientists use.

But, there are many, many other languages that are worth your time when considering

which language to use to solve a particular data science problem.

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!

I won’t dive as deeply into each of these languages, but I'll mention some notable highlights.

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 alongwith 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.

That's as far as we’ll dig into the many languages that are used to solve data science

problems.

If you have experience with a particular language, I recommend you do a web search to see what

might already be possible in terms of using it for data science.

You might be surprised at the possibilities

**Categories of Data Science Tools**

1. Data management: Process of persisting and retrieving data
2. Data integration (Extract, Transform and Load): Process of retrieving data from remote data management systems
   1. Process of loading into a local data management system part of data integration and transformation
3. Data visualition: Initial data exploration/part of final deliverable
4. Model building: Creating machine learning or deep learning model, using appropriate algorithm with a lot of data
5. Model deployment: Makes machine learning or deep learning model available to third party applications
6. Model monitoring and assessment ensures performance quality checks on models
   1. Accuracy, fairness and adversarial robustness
7. Code asset management: Versioning and other collaborative features to facilitate teamwork
8. Data Management: Same versioning and collaborative components to data
   1. Replication backup and access rights management
9. Development Environments (Integrated development environments (“IDE’s”): tools that help the Data scientist execute, test and deloy work
10. Execution environments: Tools where data repossessing, model training and deployment take place
11. Fully integrated visual tooling: Covers all previous tooling either partially or completely

Open source tools are available for various data science tasks.

In this video, we’ll have a look at the different data science tasks.

In subsequent videos we’ll walk through the most commonly used open source tools for

those tasks.

The most important tools are covered throughout this course.

Data Management is the process of persisting and retrieving data.

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.

Data Visualization is part of an initial data exploration process, as well as being part

of a final deliverable.

Model Building is the process of creating a machine learning or deep learning model

using an appropriate algorithm with a lot of data.

Model deployment makes such a machine learning or deep learning model available to third-party

applications.

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

models.

These checks are for accuracy, fairness, and adversarial robustness.

Code asset management uses versioning and other collaborative features to facilitate

teamwork.

Data asset management brings the same versioning and collaborative components to data.

Data asset management also supports replication, backup, and access right management.

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.

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

take place.

Finally, there is fully integrated, visual tooling available that covers all the previous

tooling components, either partially or completely.

This concludes this video.

In the next video we’ll start looking at open source tools for data science tasks.

**Open Source Tools for Data Science – Part 1**

1. Widely used open source Data Management tools are relational databases (need code)
   1. MySQL, PostgreSQL
   2. NoSQL databases: MongoDB, Apache, CouchDB, and Apache Cassandra
   3. File based: Hadoop File System
   4. Cloud Based: Ceph
   5. Elasticsearch: Storing data, creating search index for fast document retrieval
2. Extract, Transform and load (ETL): Task of integration and transformation in classic data warehousing world
   1. Often known as ELT (Extract, Load, Transforming): Data scientist is responsible for the data
3. Data refining and cleaning: New term for ETL
   1. Tools: Apache AirFlow (AirBNB),
   2. KubeFlow: Execute data science pipelines on top of Kubernetes
   3. Apache Kafka (LinkedIn)
   4. Apache Nifi: Nice visual editor
   5. Apache SparkSQL: Enables you to use ANSI SQL and scales up to 1000s of nodes
   6. NodeRED: provides visual editor. Consumes so little in resources it runs on small devices like Raspberry Pi
4. Open Source data visualization tools (Contain user interface)
   1. Hue: Creates visualizations from SQL queries
   2. Kibana: Data exploration and visualization web app – limited to Elastic Search
   3. Apache Superset: Data exploration and vizualisation web application
      1. Model consumable and turn it into an API (Application Programming interface)
   4. Seldon: Supports nearly every framework
      1. TensorFlow, ApacheSparkML, R and scikit-learn
      2. Can run on top of Kubernetes and Redhat Openshift
   5. MLeap: Delpoy SparkML models
   6. TensorFlow: Serve any of its models using TensorFlow service
      1. Deploy to Raspberry Pi or a smartphone using TensorFlow Lite
      2. Deploy to a web browser using TensorFlow dot JS
         1. Model Monitoring: Once machine learning model deployed, you need to keep track of its prediction performance as new data arrives to maintain outdated models
            1. ModelDB: Machine model database where info about the models are stored and can be queried

Natively supports Apache Spark, ML Pipelines and scikit-learn

* + - * 1. Prometheus: Used for machine learning model monitoring – not made specifically for this purpose
      1. Model Bias
         1. IBM AI Fairness 360 detects and mitigates against bias in machine learning models
      2. Model attacks
         1. Manipulates data or by manipulating the model itself
         2. IBM Robustness 360 Toolbox can be used to detect vulnerability to adversarial attacks and make the model more robust
         3. IBM Explainability 360 Toolkit: Makes machine learning process understble,

finding similar examples within a dataset then presented to a user for manual comparison

Illustrate training by explaining how different input variables affect the final decision of the model

Code Asset Management (Version management or version control)

Git is not the standard

Github: Provides hosting for software development version management

GitLab: Open source platform you can host and manage yourself

Bitbucket

1. Data Asset Management (data governance or data lineage): Data has to be versioned and annotated with metadata
   1. Apache Atlas
   2. ODPI Egeria: Linux Foundation – open ecosystem
      1. Offers a set of open API’s, types and interchange protocols that metadata repositories use to share and exchange data
   3. Kylo
      1. Open source data lake management software platform provides support for a wide range of data asset management tasks

In part one of this two-part series, we’ll cover data management, open source data integration,

transformation, and visualization tools.

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.

The task of data integration and transformation in the classic data warehousing world is called

ETL, which stands for “extract, transform, and load.”

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.

We’ll now introduce the most widely used open source data visualization tools.

We have to distinguish between programming libraries where you need to use code and tools

that contain a user interface.

The most popular libraries are covered in the next videos.

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.

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.

Model monitoring is another crucial step.

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.

Following are some examples of model monitoring tools:

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.

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.

Options for code asset management tools have been greatly simplified:

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.

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.

This concludes part one of this two-part series.

Now let’s move on to part two.

**Open Source Tools for Data Part 2**

1. Jupyter
   1. Python and hundreds of different programming languages through Kernels
   2. Kernels are encapsulating the different interactive interpreters for the different programming languages
   3. Jupyter can unify documentation, code, output from code, shell commands, and visualizations into a single document
2. JupyterLab: Next gen of Jupyter Notebooks – will ultimately replace Jupyter Notebooks
   1. Modern and Modular
   2. Ability to open different types of files, including Juypter Notebooks, data and terminals
   3. Can arrange files on the canvas
3. Apache Zepplin: Inspired by Jupyter Notebooks
   1. Differentiator: Integrated plotting capability
      1. Jupyter notebooks: Required to use external libraries
      2. Apache Zeppelin: Plotting doesn’t require coding. Extend these capabilities by using additional libraries
4. RStudio: 2011
   1. Exclusively runs R and all associate R libraries
   2. Python development is possible and R is tightly integrated into this tool for user experience
   3. Unifies programming, execution, debugging, remote data access, data exploration and visualization
5. Spyder: Mimic behaviour of R to bring functionality to Python world
   1. Does not have same level of R
   2. Python world Jupyter is prominent

Cluster-Computing Frameworks: When data doesn’t fit into a single computer’s storage

* + - 1. ApacheSpark: Used across all industries (fortune 500)
         1. Linear scalability: Double the number of servers in a cluster, you’ll double its performance
      2. Apache Flink:
         1. Difference is Spark is a batch data processing engine, capable of processing huge amounts of data file by file
         2. Flink is a stream processing image, focus on processing real-time data streams
         3. Spark is usually the choice
      3. Ray: Focus on deep learning model training

Fully Integrated and Visual Tools: No programming necessary

1. Knime: University of Konstanz in 2004
   1. Visual user interface with drag-and-drop capabilities
   2. Can be extended by programming in R and Python, connects to Apache Spark
2. Orange
   1. Less flexible than Knime but easier to use

Welcome to part two of this series.

In this section, we’ll cover the development environment, open source data integration,

transformation, and visualization tools.

One of the most popular current development environments that data scientists are using

is “Jupyter.”

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.

This diagram shows how Spyder integrates code, documentation, visualizations, and other components

into a single canvas.

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.

In this video, you’ve learned about the most common data science tasks and which open

source tools are relevant to those tasks.

In the next video, we’ll describe some established commercial tools that you’ll encounter in

your data science experience.

Let’s move on to the next vi

**Commercial Tools for Data Science**

1. Data Management: Most relevant data stored in
   1. An oracle database, Microsoft SQL Server, IBM Db2
   2. These are industry standards
2. Commercial Supports delivered directly from software vendors, influential partners and support networks
   1. Extract, transform and load (ETL) tools
   2. Leaders: SAP, Oracle, SAS, Talend, Microsoft
   3. Support design and deployment of ETL through graphical interface
   4. Provide connectors to most of the commercial and open source target information systems
   5. Watson Studio Desktop: Includes Data Refinery – enables the defining and execution of data integration processes in a spreadsheet style
3. Commercial Environments: Data visualizations using business intelligence (BI) tools
   1. Create visually attractive, easy-to-understand reports and live dashboards
   2. Tableau, Microsoft Power BI, and IBM Cognos Analytics
4. Type of Vizualition that targets Data Scientists
   1. Watson Studio Desktop
5. To build a machine learning model using a commercial tool: use data mining product
   1. SPSS Modeler and SAS enterprise miner (also in Watson studio desktop)
6. Commercial Software can export in open format
   1. SPSS. Modeler supports Predictive Model Markup Language (PMML)
7. Model Monitoring
   1. New discipline and no relevant commercial tools available
   2. Open source is first choice
8. Code Asset Management
   1. New discipline
   2. Open source with Git and GitHub is the effective standard
9. Data Asset Management: AKA Data Governance or Data Lineage
   1. Data must be versioned and annotated using metadata
   2. Crucial part of data science
   3. Venders including Informica Enterprise Data Governance and IBM provide tools
      1. IBM Infosphere Information Governance Catalog
         1. Data dictionary, facilitates discovery of data assets
10. Data Asset
    1. Each data asset is assigned to a data steward – the data owner
    2. Data owner is responsible and can be contacted
    3. Data lineage also covered – allowing a user to track transformation steps followed creating data assets
    4. Data lineage includes reference to actual source data
    5. Rules and policies added to reflect complex regulatory and business requirements for data privacy and retention
    6. Watson Studio is a fully integrated development environment for data scientists
       1. Usually consumed through the cloud
    7. Watson desktop combines Jupyter Notebooks with graphical tools to maximize performance.
    8. Watson studio and Watson Open Scale is a fully integrated tool cover the full data science life cycle and all the tasks
       1. These can be deployed in a local data center on top of Kubernetes or RedHat OpenShift
       2. H2O Driverless AI covers complete data science life cycle

We previously covered open source tools for data science.

Now, let’s look at the commercial options you’ll find in many enterprise projects.

Let’s revisit our overview of different tool categories.

In 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.

When we focus on commercial data integration tools, 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.

In the commercial environment, 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.

We’ll talk more about cloud-based tools in the next video.

In commercial software, 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.

Model monitoring is a new discipline and there are currently no relevant commercial tools

available.

As a result, open source is the first choice.

The same is true for code asset management.

Open source with Git and GitHub is the effective standard.

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.

We’ll talk more about both in the next lesson.

but 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.

In this lesson, you’ve learned how most common data science tasks are supported by

commercial tools.

In the next video, we’ll discover data science tools that are available exclusively on the

cloud.

**Cloud Based Tools for Data Science**

1. Platform
   1. Multiple server machines
   2. Watson Studio and Watson OpenScale: covers development cycle for data science, machine leaning and AI tasks
   3. Microsoft Azure Machine Learning: All data science, machine learning and AI tasks
   4. H2O Driverless Ai: download and install. Not strictly Platform or Software as a service (PaaS or SaaS)
   5. SaaS: Cloud provider operates the tool for you in the cloud
   6. Cloud provider backs up data, configuration and installs updates
   7. Amazon Web Service DynamoDB
      1. NoSQL database allows storage and retrieval
      2. JSON document structure
   8. Cloudant
      1. Basically Apache CouchDB – migrated to another server without changing application
   9. IBM offers DB2
      1. SaaS – taking operational tasks away from user
2. Commercial Data Integration Tools
   1. ELT: Extract, Load, Transform
   2. Transformation not done by data integration team, but by the data scientist
   3. Informatica Cloud Data Integration and IBM’s Data Refinery
      1. Transform large data into spreadsheet like user interface
      2. Part of IBM Watson
   4. DataMeer: Smaller vendor
   5. Watson Studio visualizations: 3D bar chart, Colouring, hierarchical edge bundling, classic bar chart, 2D scatter plot with heat map, tree map (distribution of subsets within a set), pie chart, word cloud
   6. Watson Machine learning: Train and build models using open source libraries
   7. Google has AI Platform Training
   8. SPSS: deploy any type of asset created by SPSS software tool suite
      1. Exports models like Predictive Model Markup Language (PMML)
      2. Can be read on numerous commercial and open packages
   9. Watson Machine learning can deploy a model using REST interface
   10. Amazon SageMaker Model Monitor continuously monitors deployed machine learning and deep learning models – Cloud tool
   11. Watson Open Scale and Studio unify the landscape
   12. Integration allows us to use the same tools for multiple tasks

Since we’ve previously covered open source tools for data science, let’s look at the

commercial options you’ll find in many enterprise projects.

Take another look at the overview of different tool categories.

Since cloud products are a newer species, they follow the trend of having multiple tasks

integrated in tools.

This especially holds true for the tasks marked green in the diagram.

Let’s start with the fully integrated visual tools category.

Since these tools introduce a component where large scale execution of data science workflows

happens in compute clusters, we’ve changed the title here and added the word “Platform.”

These clusters are composed of multiple server machines, transparently for the user, in the

background.

Watson Studio, together with Watson OpenScale, covers the complete development life cycle

for all data science, machine learning, and AI tasks.

Another example is Microsoft Azure Machine Learning.

This is also a fully cloud-hosted offering supporting the complete development life cycle

of all data science, machine learning, and AI tasks.

And finally, another example is H2O Driverless AI, which we’ve already introduced in the

last video.

Although it is a product that you download and install, one-click deployment is available

for the common cloud service providers.

Since operations and maintenance are not done by the cloud provider, as is the case with

Watson Studio, Open Scale, and Azure Machine Learning, this delivery model should not be

confused with Platform or Software as a Service -- PaaS or SaaS.

In data management, with some exceptions, there are SaaS versions of existing open source

and commercial tools.

Remember, SaaS stands for “software as a service.”

It means that the cloud provider operates the tool for you in the cloud.

As an example, the cloud provider operates the product by backing up your data and configuration

and installing updates.

As mentioned, there is proprietary tooling, which is only available as a cloud product.

Sometimes it’s only available from a single cloud provider.

One example of such a service is Amazon Web Services DynamoDB, a NoSQL database that allows

storage and retrieval of data in a key-value or a document store format.

The most prominent document data structure is JSON (pronounced “jay-sun”).

Another flavour of such a service is Cloudant, which is a database-as-a-service offering.

But, under the hood it is based on the open source Apache CouchDB.

It has an advantage: although complex operational tasks like updating, backup, restore, and

scaling are done by the cloud provider, under the hood this offering is compatible with

CouchDB.

Therefore, the application can be migrated to another CouchDB server without changing

the application.

And IBM offers Db2 as a service as well.

This is an example of a commercial database made available as a software-as-a-service

offering in the cloud, taking operational tasks away from the user.

When it comes to commercial data integration tools, we talk not only about “extract,

transform, and load,” or “ETL” tools, but also about “extract, load, and transform,”

or “ELT,” tools.

This means the transformation steps are not done by a data integration team but are pushed

towards the domain of the data scientist or data engineer.

Two widely used commercial data integration tools are Informatica Cloud Data Integration

and IBM’s Data Refinery.

Data Refinery enables transformation of large amounts of raw data into consumable, quality

information in a spreadsheet-like user interface.

Data Refinery is part of IBM Watson Studio.

The market for cloud data visualization tools is huge, and every major cloud vendor has

one.

An example of a smaller company’s cloud-based data visualization tool is DataMeer.

IBM offers it’s famous Cognos Business intelligence suite as cloud solution as well.

IBM Data Refinery also offers data exploration and visualization functionality in Watson

Studio.

Again, these are just some examples of a rapidly changing and growing commercial ecosystem

among a huge number of established and emerging vendors.

In Watson Studio, an abundance of different visualizations can be used to better understand

data.

For example, this 3D bar chart enables you to visualize a target value on the vertical

dimension, which is dependent on two other values on the horizontal dimensions.

Coloring enables you to visualize a third dimension.

Hierarchical edge bundling enables you to visualize correlations and affiliations between

entities.

If sufficient, a classic bar chart can do the job as well, whereas a 2D scatter plot

with a heat map shows two dependent data fields, one on the y axis and one as color intensity.

A tree map shows distribution of subsets within a set, the famous pie chart does the same

but in a non-hierarchical manner, and finally, a word cloud pops out significant terms in

a document corpus.

Model building can be done using a service such as Watson Machine Learning.

Watson Machine Learning can train and build models using various open source libraries.

Google has a similar service on their cloud called AI Platform Training.

Nearly every cloud provider has a solution for this task.

Model deployment in commercial software is usually tightly integrated to the model building

process.

Here is an example of the SPSS Collaboration and Deployment Services, which can be used

to deploy any type of asset created by the SPSS software tools suite.

The same holds for other vendors.

In addition, commercial software can export models in an open format.

As an example, SPSS Modeler supports exporting models as Predictive Model Markup Language,

or “PMML,” which can be read by numerous other commercial and open software packages.

Watson Machine Learning can also be used to deploy a model and make it available to consumers

using a REST interface.

Amazon SageMaker Model Monitor is an example of a cloud tool that continuously monitors

deployed machine learning and deep learning models.

Again, every major cloud provider has similar tooling.

This is also the case for Watson OpenScale.

OpenScale and Watson Studio…

…unify the landscape.

Everything marked in green can be done using Watson Studio and Watson OpenScale.

We’ll cover Open Scale will be covered in a later video.

You’ve learned how the most common tasks in data science are supported by commercial

cloud tools.

Integration provides us the ability to use the same tools for multiple tasks.

In the next videos, we’ll look at packages, APIs, datasets, and models for data science

**Libraries for Data Science**

Libraries In Python

* 1. Collection of functions and methods that enable you to perform a wide variety of actions without writing the code yourself
  2. Frameworks: Built-in modules providing different functionality

Scientific Computing Libraries

1. Pandas
   1. Data structures and tools for effective data cleaning, manipulation and analysis
   2. Dataframe: two dimensional table consisting of columns and rows
      1. Provide easy indexing

Visualization Libraries

1. Numpy
   1. Based on arrays, enabling mathematical functions added to these arrays
   2. Pandas is actually built on top of Numpy
2. Matplotlib
   1. Most well known, excellent for graphs and plots
3. Seaborn
   1. Based on Matplotlib, generate heat maps, time series, and violin plots

Machine Learning Deep learning

1. Scikit-learn
   1. Machine learning
   2. Statistical modeling (regression, classification, clustering etc)
   3. Built on Numpy, SciPy and Matplotlib
2. Keras
   1. Deep Learning
   2. Build standard deep learning models
   3. Functions using Graphics Processing Unit (GPU)
3. TensorFlow
   1. Low-level framework, used in large production of deep learning models
   2. Not great for experimentation
4. Pytorch
   1. Used for experimentation – easy to test ideas
5. Apache Spark
   1. Gerneral purpose cluster comuputing framework: Process data using computer clusters
   2. Process data in parallel using multiple computers simutaeously
   3. Can use Python, R, Scala or SQL for processing jobs

Spark Libraries

* + - 1. Vegas: Statistical data vizualition
      2. BigDl: Deep learning

R Libraries

1. Ggplot2
2. Keras
3. Tensorflow
4. Now superseded by Python

In this video, we will review several data science libraries.

Libraries are a collection of functions and methods that enable you to perform a wide

variety of actions without writing the code yourself.

We will focus on Python libraries: Scientific Computing Libraries in Python

Visualization Libraries in Python High-Level Machine Learning and Deep Learning

Libraries – “High-level” simply means you don’t have to worry about details, although

this makes it difficult to study or improve Deep Learning Libraries in Python

Libraries used in other languages

Libraries usually contain built-in modules providing different functionalities that you

can use directly; these are sometimes called “frameworks.”

There are also extensive libraries, offering a broad range of facilities.

Pandas offers data structures and tools for effective data cleaning, manipulation, and

analysis.

It provides tools to work with different types of data.

The primary instrument of Pandas is a two-dimensional table consisting of columns and rows.

This table is called a “DataFrame” and is designed to provide easy indexing so you

can work with your data.

NumPy libraries are based on arrays, enabling you to apply mathematical functions to these

arrays.

Pandas is actually built on top of NumPy

Data visualization methods are a great way to communicate with others and show the meaningful

results of analysis.

These libraries enable you to create graphs, charts and maps.

The Matplotlib package is the most well-known library for data visualization, and it’s

excellent for making graphs and plots.

The graphs are also highly customizable.

Another high-level visualization library, Seaborn, is based on matplotlib.

Seaborn makes it easy to generate plots like heat maps, time series, and violin plots.

For machine learning, the Scikit-learn library contains tools for statistical modeling, including

regression, classification, clustering and others.

It is built on NumPy, SciPy, and matplotlib, and it’s relatively simple to get started.

For this high-level approach, you define the model and specify the parameter types you

would like to use.

For deep learning, Keras enables you to build the standard deep learning model.

Like Scikit-learn, the high-level interface enables you to build models quickly and simply.

It can function using graphics processing units (GPU), but for many deep learning cases

a lower-level environment is required.

TensorFlow is a low-level framework used in large scale production of deep learning models.

It’s designed for production but can be unwieldy for experimentation.

Pytorch is used for experimentation, making it simple for researchers to test their ideas

Apache Spark is a general-purpose cluster-computing framework that enables you to process data

using compute clusters.

This means that you process data in parallel, using multiple computers simultaneously.

The Spark library has similar functionality as

Pandas Numpy

Scikit-learn

Apache Spark data processing jobs can use Python

R Scala, or SQL

There are many libraries for Scala, which is predominately used in data engineering

but is also sometimes used in data science.

Let’s discuss some of the libraries that are complementary to Spark

Vegas is a Scala library for statistical data visualizations.

With Vegas, you can work with data files as well as Spark DataFrames.

For deep learning, you can use BigDL.

R has built-in functionality for machine learning and data visualization, but there are also

several complementary libraries: ggplot2 is a popular library for data visualization

in R. You can also use libraries that enable you

to interface with Keras and TensorFlow.

R has been the de-facto standard for open source data science but it is now being superseded

by Python.

**Application Programming Interfaces API’s**

API: Lets two pieces of data talk to each other

e.g. Panda’s API component

Separate API’s written in Python, Java, C++, Java and Go

1. TensorFlow
   1. Julia, Matlab, R and Scala
2. Rest API’s
   1. Communicate using the internet, take advantage of storage, data access, AI algorithms and more
   2. Representational State Transfer
   3. Program is called the client – API’s used to interact with web services
   4. Client: You/code
   5. Webservice: Service
   6. Endpoint: Client finds the service through the endpoint
   7. HTTP: Transmitting data over the internet (JSON, Request) through an API
   8. Watson Speech to text API
      1. Post-request sent to Watson API
      2. Waston is making a get request

In this video we will discuss Application Programming Interfaces, or APIs. Specifically,

we will discuss: What an API is, API Libraries, REST APIs, including:

Request and Response. An API lets two pieces of software talk to each other. For example

you have your program, you have some data, you have other software components. You use

the API to communicate with the other software components.You don’t have to know how the

API works, you just need to know its inputs and outputs. Remember, the API only refers

to the interface, or the part of the library that you see. The “library” refers to

the whole thing. Consider the pandas library. Pandas is actually a set of software components,

many of which are not even written in Python. You have some data.

You have a set of software components. We use the pandas API to process the data

by communicating with the other software components. There can be a single software component at

the back end, but there can be a separate API for different languages. Consider TensorFlow,

written in C++. There are separate APIs in Python, JavaScript,

C++ Java, and Go. The API is simply the interface.

There are also multiple volunteer-developed APIs for TensorFlow; for example Julia, MATLAB,

R, Scala, and many more. REST APIs are another popular type of API. They enable you to communicate

using the internet, taking advantage of storage, greater data

access, artificial intelligence algorithms, and many other resources. The RE stands for

“Representational,” the S stands for “State,” the T stand for “Transfer.” In rest APIs,

your program is called the “client.” The API communicates with a web service that you

call through the internet. A set of rules governs Communication, Input or Request, and

Output or Response. Here are some common API-related terms. You or your code can be thought of

as a client. The web service is referred to as a resource. The client finds the service

through an endpoint. The client sends the request to the resource and the response to

the client. HTTP methods are a way of transmitting data over the internet We tell the REST APIs

what to do by sending a request. The request is usually communicated through an HTTP message.

The HTTP message usually contains a JSON file, which contains instructions for the operation

that we would like the service to perform. This operation is transmitted to the web service

over the internet. The service performs the operation. Similarly, the web service returns

a response through an HTTP message, where the information is usually returned using

a JSON file. This information is transmitted back to the

client. The Watson Speech to Text API is an example

of a REST API. This API converts speech to text. In the API call, you send a copy of

the audio file to the API; this process is called a post request. The API then sends

the text transcription of what the individual is saying. The API is making a get request.

The Watson Language-Translator API provides another example. You send the text you would

like to translate into the API, the API translates the text and sends the translation back to

you. In this case we translate English to Spanish. In this video, we’ve discussed

what an API is, API Libraries, REST APIs, including Request and Response. Thank you

for watching this video.

**Data Sets - Powering Data Science**

Collection of data is a body of structured data: Text, numbers or media

1. Tabular
   1. Rows and colums that store data
   2. CSV tabular format (Comma Separated Values)
   3. Delimited text file, each line represents a row and data values separated by a comma
2. Hierarchical Data/Network data
   1. Represent relatioships between data
   2. Organized in a tree like structure
   3. Network data stored as a graph
   4. Raw data files: E.g. MNIST data set – handwritten digits to train processing systems
3. Private data
   1. Confidential
   2. Private or personal info
   3. Commercially sensitive
4. Open data
   1. Scientific institutions, governments, organizations, companies make data public

Where to Find Open Data

1. Open data portal list (datacatalogs.org) – open knowledge foundation
2. Governmental
   1. Data.un.org/ data.gov/ europeandataportal.eu/en/
3. Kaggle (Kaggle.com/datasets)
4. Google (datasetsearch.research.google.com)

Community Data License Agreement

1. Linux foundation CDLA (Community data license agreement)
   1. Collaborative licenses to enable access, sharing and use of data openly among individuals and organizations
   2. Cdla.io) CDLA-sharing: Permission to use and modify data; publication only under same terms as original data
   3. CDLA-permissive: Permission to use and modify data; no obligations
      1. No restrictions to conclusions you may derive

In this video we’ll discuss data sets: what they are, why they are important in data science,

and where to find them.

Let’s first loosely define what a data set is.

A data set is a structured collection of data.

Data embodies information that might be represented as text, numbers, or media such as images,

audio, or video files.

A data set that is structured as tabular data comprises a collection of rows, which in turn

comprise columns that store the information.

One popular tabular data format is "comma separated values," or CSV.

A CSV file is a delimited text file where each line represents a row and data values

are separated by a comma.

For example, imagine a data set of observations from a weather station.

Each row represents an observation at a given time, while each column contains information

about that particular observation, such as the temperature, humidity, and other weather

conditions.

Hierarchical or network data structures are typically used to represent relationships

between data.

Hierarchical data is organized in a tree-like structure, whereas network data might be stored

as a graph.

For example, the connections between people on a social networking website are often represented

in the form of a graph.

A data set might also include raw data files, such as images or audio.

The MNIST dataset is popular for data science.

It contains images of handwritten digits and is commonly used to train image processing

systems.

Traditionally, most data sets were considered to be private because they contain proprietary

or confidential information such as customer data, pricing data, or other commercially

sensitive information.

These data sets are typically not shared publicly.

Over time, more and more public and private entities such as scientific institutions,

governments, organizations and even companies have started to make data sets available to

the public as “open data," providing a wealth of information for free.

For example, the United Nations and federal and municipal governments around the world

have published many data sets on their websites, covering the economy, society, healthcare,

transportation, environment, and much more.

Access to these and other open data sets enable data scientists, researchers, analysts, and

others to uncover previously unknown and potentially useful insights.

They can create new applications for both commercial purposes and the public good.

They can also carry out new research.

Open data has played a significant role in the growth of data science, machine learning,

and artificial intelligence and has provided a way for practitioners to hone their skills

on a wide variety of data sets.

There are many open data sources on the internet.

You can find a comprehensive list of open data portals from around the world on the

Open Knowledge Foundation’s datacatalogs.org website.

The United Nations, the European Union, and many other governmental and intergovernmental

organizations maintain data repositories providing access to a wide range of information.

On Kaggle, which is a popular data science online community, you can find and contribute

data sets that might be of general interest.

Last but not least, Google provides a search engine for data sets that might help you find

the ones that have particular value for you.

It’s important to recognize that open data distribution and use might be restricted,

as defined by its licensing terms.

In absence of a license for open data distribution, many data sets were shared in the past under

open source software licenses.

These licenses were not designed to cover the specific considerations related to the

distribution and use of data sets.

To address the issue, the Linux Foundation created the Community Data License Agreement,

or CDLA.

Two licenses were initially created for sharing data: CDLA-Sharing and CDLA-Permissive.

The CDLA-Sharing license grants you permission to use and modify the data.

The license stipulates that if you publish your modified version of the data you must

do so under the same license terms as the original data.

The CDLA-Permissive license also grants you permission to use and modify the data.

However, you are not required to share changes to the data.

Note that neither license imposes any restrictions on results you might derive by using the data,

which is important in data science.

Let’s say, for example, that you are building a model that performs a prediction.

If you are training the model using CDLA-licensed data sets, you are under no obligation to

share the model, or to share it under a specific license if you do choose to share it.

In this video you’ve learned about open data sets, their role in data science, and

where to find them.

We’ve also introduced the Community Data License Agreement, which makes it easier to

share open data.

One important aspect that we didn’t cover in this video is data quality and accuracy,

which might vary greatly depending on who collected and contributed the data set.

While some open data sets might be good enough for personal use, they might not meet enterprise

requirements due to the impact they might have on the business.

In the next module, you will learn about the Data Asset eXchange, a curated open data repository.

**Sharing Enterprise Data – Data Asset eXchange**

Difficult to find data sets that are high quality and have clearly defined data usage and terms

The IBM Data Asset eXchange

1. Trusted source for finding data sources ready to use in enterprise applications
2. Multiple application domains (images etc)
3. High level of duration and lisencing and usage terms – easy to adopt
4. Data friendly lisences
5. Single placed asset unique datasets
6. Notebooks and tutorials
   1. Complex analysis, charts, statistical analysis, time-series analysis, training machine learning models and integrating deep learning via Model Asset eXchange
7. End-to-end analytic and machine learning workflows
8. Consume open data under clearly defined lisence terms

Using the Data Sets

1. Download compressed data set archive from cloud storage
2. Explore the data set using Juypter notebooks
   1. Review the metadata (licensing, format and size)
   2. Preview some parts of the data set
3. Complented by one or more Juypter notebooks
   1. Used to explore cleaning preproseing and preporatory analysis

Despite the growth of open data sets that are available to the public, it can still

be difficult to discover data sets that are both high quality and have clearly defined

license and usage terms.

To help solve this challenge, IBM created the Data Asset eXchange, or "DAX,”, which

we’ll introduce in this video.

DAX provides a trusted source for finding open data sets that are ready for to use in

enterprise applications.

These data sets and which cover a wide variety of domains, including images, video, text,

and audio.

Because DAX provides a high level of curation for data set quality, as well as licensing

and usage terms, DAX data sets are typically easier to adopt, whether in research or commercial

projects.

Wherever possible, DAX aims to make data sets available under one of the variants of the

CDLACommunity Data License Agreement, in order to foster data sharing and collaboration.

DAX also provides a single place to access unique data sets, in particular from IBM Research

projects.

To make it easier for developers to get started with using the data sets, DAX also provides

tutorials in the form of notebooks that walk through the basics of data cleaning, pre-processing,

and exploratory analysis.

For some data sets, there are also notebooks illustrating how to perform more complex analysis,

such as creating charts, statistical analysis, time-series analysis, training machine learning

models, and integrating deep learning via using the Model Asset eXchange, (a project

closely related to DAX and also available on the IBM Developer website).

In this way, DAX helps developers to create end-to-end analytic and machine learning workflows

and to consume open data and models with confidence under clearly defined license terms.

Let’s say you’ve found a data set that might be of interest to you.

On the data set page you can download the compressed data set archive from cloud storage,

explore the data set using Jupyter Notebooks, review the data set metadata, such as format,

licensing terms and size, and preview some parts of the data set.

Most data sets on DAX are complemented by one or more Jupyter Notebooks that you can

use to perform data cleaning, pre-processing, and exploratory analysis.

These notebooks run "as is"as is in Watson Studio, IBM’s Data Sciencedata science platform.

Jupyter Notebooks and Watson Studio are covered later during in this course.

In this video, you’ve learned about IBM’s open data repository, the Data Asset eXchange.

In the hands-on lab you’ll have a chance to explore the repository.

**Machine Learning Models**

What is a Model?  
1. Data contain a wealth of info

1. Machine Learning used algorithms (aka models) to identify patterns in the data
   1. The process of learning these patterns is called model training
   2. It tries to make predictions from the patterns based on the patterns it has learned past data

3 classes of Machine Learning

* + - 1. Supervised Learning
         1. Human provides input data and correct outputs – model tries to identify relationships and dependencies between the input data and the correct output
         2. Used to solve regression and classification models
         3. Regression (home sales, stok market prices
         4. Classification (spam emails, fraud detection, etc)
      2. Unsupervised learning
         1. Not labeled by a human
         2. Model analysies data and patterns and structures based only on the data itself
         3. Clustering and anomaly detection
         4. Clustering: shopping behaviour based on content of a shopping cart
         5. Anomaly: Identifies outliers, fraudulent transactions or suspicious log in attempts
      3. Reinforcement learning
         1. Conceptually similar to human learning processes
         2. Learns the best set of actions to take, given its environment, to get the most rewards over time
         3. Successful in beating humans at chess

Deep Learning

1. Tries to loosely emulate how the human brain works
2. Applications
   1. Natural language processing
   2. Image, audio and video analysis
   3. Time series forecasting
3. Requires very large data sets of labeled data sets to train a model and is compute intensive – requires special purpose hardware to achieve training times
4. Can be built from scratch or downloaded from a public model repository
5. Built using frameworks
   1. Tensorflow, PyTorch, Keras
6. Typically a Python API
7. Model zoos: Popular model repositories
   1. ONNX model zoo as well as those above
   2. Academic research groups

Building a model using high-level tasks

1. Collect and prepare data
   1. Label raw data (drawing bounding boxes around objects and labeling them)
   2. Build model or select a model
   3. Train the model on the prepared data (learns how to identify data)
   4. Analyze training results and repeat the process till the model meets performance criteria
   5. DEPLOY TO MAKE AVAIALBLE TO APPLICATIONS

n this video, we’ll introduce you to 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.

In this video, you’ve learned about machine learning and deep learning, what they are

used for, and where to find open source models.

In the next video, we’ll introduce you to the Model Asset eXchange, a curated collection

of ready-to-use and customizable deep learning models.

**Model Asset Exchange**

MAX

1. Free open-source deep learning microservices – cuts down on time to value
   1. Use pre-trained or custom trainable state of the art models to solve custom business models
   2. Fully tested, deployed in deep learning or cloud environments
   3. Approved for personal and commercial use
2. Available for a variety of domains:
   1. Object detection (which objects are in this image)
   2. Image, audio and text classification (what is in this…)
   3. Named entity recognition (Identify entities in text)
   4. Image to text translation (generate image caption)
   5. Human pose detection

Max Model-Serving Microservice

1. Pretrained deep learning model
2. Code that preprocesses the input before it is analyzed by the model
3. Code that postprocesses the model output
4. A standardized public API that makes the services functionality vailable to applications
   1. Distributed as Docker images: Docker is a container platform that makes it easy to build apps and to deploy them development test or production environment
   2. Published on Github and be downloaded and customized as needed and be used in commercial or personal environments
      1. Deploy and run using Kubernetes – open-source system for automating deployment, scaling and management
      2. Private, hybrid, or public clouds
   3. Red Hat OpenShift
      1. IBM cloud, Google Cloud Platform, Amazon Web Services and Microsoft Azure

Model-Serving Microservice API

* + - 1. Model-serving microservices expose standardized REST API’s
      2. API exposes a prediction endpoint and one or more metadata endpoints

Max Object Detector

1. /model/predict: takes an image as input and returns a list o objects detected along with the bounding box coordinates of where the object is located
2. /Model/lables and /model/metadata: objects that can be detected and the deep learning model used to derive the answer given the input
3. Application-friendly inputs and outputs (JSON)

In this video, we will introduce you to the Model Asset eXchange on IBM Developer, a free

open source resource for deep learning models.

Throughout the video we will refer to the Model Asset eXchange as "MAX."

In the previous video, we briefly outlined the high-level tasks you need to complete

to train a model from scratch.

Due to the amount of data, labor, time, and resources required to complete the tasks,

time to value can be quite long.

To reduce time to value, consider taking advantage of pre-trained models for certain types of

problems.

These pre-trained models can be ready to use right away, or they might take less time to

train.

The Model Asset eXchange is a free open source repository for ready-to-use and customizable

deep learning microservices.

These microservices are configured to use pre-trained or custom-trainable state-of-the-art

deep learning models to solve common business problems.

These models have been reviewed, tested, and can be quickly deployed in local and cloud

environments.

All models in MAX are available under permissive open source licenses, making it easier to

use them for personal and commercial purposes and reducing the risk of legal liabilities.

On MAX, you can find models for a variety of domains, including image, audio, video,

and natural language analysis.

This list includes a small selection.

In the lab for this module, you’ll have a chance to explore those models.

Let’s take a look at the components of a typical model-serving microservice.

Each microservice includes the following components:

A pre-trained deep learning model.

Code that pre-processes the input before it is analyzed by the model and code that post-processes

the model output.

A standardized public API that makes the services’ functionality available to applications.

The MAX model-serving microservices are built and distributed as open-source Docker images.

Docker is a container platform that makes it easy to build applications and to deploy

them in a development, test, or production environment.

The Docker image source is published on GitHub and can be downloaded, customized as needed,

and used in personal or commercial environments.

You can deploy and run these images in a test or production environment using Kubernetes,

an open-source system for automating deployment, scaling, and management of containerized applications

in private, hybrid, or public clouds.

A popular enterprise-grade Kubernetes platform is Red Hat OpenShift, which is available on

IBM Cloud, Google Cloud Platform, Amazon Web Services, and Microsoft Azure.

The model-serving microservices expose a REST API that developers can use to incorporate

deep learning into their applications and services.

Because REST APIs can be consumed using any programming language, you can easily integrate

these services into your existing ecosystem.

The API exposes a prediction endpoint and one or more metadata endpoints.

This example shows the endpoints for the Object Detection microservice.

The /model/predict endpoint takes an image as input and returns as a response a list

of objects that were detected in the image, along with bounding box coordinates that identify

where the detected object is located.

Some prediction endpoints can also accept additional input parameters that impact the

produced results, such as filters.

This microservice exposes two metadata endpoints, /model/labels and /model/metadata.

These endpoints provide information such as the objects that can be detected and the deep

learning model used to derive the answer given the input.

In the lab portion of this module, you will have a chance to explore and test these endpoints

using a web browser.

Each endpoint accepts application-friendly inputs, such as an image in JPG, PNG, or GIF

format, instead of a model-specific data structure.

Each endpoint also generates application-friendly outputs, such as standardized JSON, which

is a lightweight data-interchange format.

Let’s take a closer look at what happens when an application invokes the prediction

endpoint.

In this example, a user has selected an image in a web application, the prediction endpoint

is invoked, and the image is uploaded.

The microservice prepares the input image for processing, runs the deep learning model

that identifies objects in the image, generates a response using the prediction results, and

returns the result to the application.

The application renders the results by drawing bounding boxes and labels.

In this video, we’ve introduced the Model Asset eXchange, a free and open source repository

for microservices that make deep learning functionality available to applications and

services in local and cloud environments.

In the lab, you will have a chance to try a model-serving microservice, explore its

API, and learn more about how you can leverage it from a web application and an Internet

of Things application

**Other Tools for Data Science**

1. Watson Knowledge catalogue
2. Data Refinery: graphical tools for analyzing and preparing data
3. SPSS products (modeler flows in Watson Studio): easy to use graphical interfaces
   1. IBM SPSS Statistics and IBM SPSS Modeler
   2. Statistical and machine learning algorithms and data transformations
4. Model deployment
   1. Open standards and Watson machine learning
5. AutoAI
   1. Automatically computes the best data pipeline
6. Watson openscale
   1. Helps ensure fairness and explainability of the models

**Watson knowledge Catalogue**

1. Units all information assets into a single metadata rich catalogue
   1. Based on wtsons undertaning of reationships between assets and how they are being used and socialized among users and different projects
2. Watson knowledge catalogue is a data calagoue that I integrated with an enterprise data governance platform.
3. It also merges the analytics capabilities of Watson Studio
4. Main Features;
   1. Find data, catalogue data, govern data, understand data, power data science, prepare data, connect data, depoy anywhere
5. Protects data from misuse and enables the sharing of assets with automated, dynamic masking of sensitive data elements
6. Can interactively discover, cleanse and prepare data with a buit-in data refinery
7. Conections to more thn 30 IBM and third party sources
8. Can be run anywhere with cloud pack for data
   1. Fully integrated AI and dta built on red hat containership openbase
   2. Deployed easily anywhere
9. Catalogue contains metadata about the contents of assets and how to access them
10. Stored in an encryted IBM coud object storage instance
    1. You can upload and specify any data
    2. The split between the metabata and where it is actually stored means that you can keep your data wherever it is. You don’t need to move it into the catalogue. The catalogue only contains metadata.
    3. You can have the data in on premesis data repositories or in other IBM cloud services
       1. IBM Cloudant
       2. IBM db2 on cloud
       3. IBM db2 warehouse on cloud
    4. Or non IBM coud services like
       1. Amazon web service
       2. Microsoft Azure
    5. Streaming data services
       1. Twitter
    6. Dark data sources
       1. PDFs
11. Included in the metadata is how to access the data asset
    1. Location and credentials

**Data Refinery**

* + - 1. Cleansing, shaping and preparing data takes time
      2. Get in the way of analyzing the data and building ML models
      3. Data sets typically not in a format reaily consumable. They must be refined and cleansed
      4. IBM data Refinery simplifies these tasks with an interactive visual interface that enables self-service data preparation
      5. Data refinery comes with Watson studio – on public/private cloud and desktop

**SPSS Modeler Flows in Watson Studio**

1. Data management capabilities, as well as tools for data preparation, visualization and model building and model deployment
2. Integral Solutions Ltd. 1994. Orginally called Clementin
3. 1998 aquired by 1998 by SPSS
   1. 2009 aquired by IBM
4. Data mining and text analytics software application
5. Visual interface to leverage statistical and data mining algorithms without programing
6. Complex predictive modeling pipeline that are easily accessible
7. Pentagons: Modeling nodes
8. Churn: predictive on who will leave a service
9. Hexagons: Specify roles, target, preditor or none
   1. Continous, nominal or flag for all variables
10. Flag: variable with two categories Positive or Negative
11. Yellow Nugget: Filter out variables that are not preictors for the target
12. Data audit node: Shows various properties of the data
    1. Number of outliers and percentage of valid values
    2. Help to create a node for missing value imputation (replacing missing values of variables based on domain knowledge)
13. Supernodes: imputing missing values – shaped as a star
14. Finally attatch logistic regression model node and click run
    1. Another model nugget appears
15. Always use a partition node to hold a subset of records for the purpose of testing and validation
16. In the model setup screen select use partitioned data
    1. This will help detect model over-fitting
17. Scores: yellow model nugget can be used to compute predictions
    1. Connect data source in question to the nugget and create an output to a table
18. Analysis node: wil compute some model computer metrics

In this lesson we will discuss two products that are very helpful for data

scientists. Both came to IBM with the SPSS acquisition in 2009. First is IBM

SPSS Modeler. Let's review the different tool categories we discussed previously.

IBM SPSS Modeler includes data management capabilities and tools for

data preparation, visualization, model building and model deployment. The

product was created by Integral Solutions Limited in the United Kingdom

in 1994 and was originally called Clementine. It was acquired by a company

called SPSS in 1998 and SPSS was in turn acquired by IBM in 2009. SPSS Modeler is

a data mining and text analytics software application. It's used to build

predictive models and conduct other analytics tasks. It has a visual

interface that enables users to leverage statistical and data mining algorithms

without programming. One of its main goals from the beginning was to create

complex predictive modeling pipelines that are easily accessible. A sample

modeler stream shown here includes one round data source node, three triangular

graph nodes, one hexagonal node for computing, a new variable, and a square

node for an output table. Below the canvas, we can see the rich node palette

with separate tabs for data sources, record in field operations, graphs, models,

output and so on. Nodes and different tabs have different

shapes with Pentagon's used for modeling nodes. Let's examine the sample stream

that comes as an example with the product. It starts with a data set of

telecommunications records and the goal is to build a model to predict which

customers are about to leave the service otherwise known as churn. The data source

is shown by the round node on the left side, a hexagon type node typically follows

a data source node and it enables us to

specify roles, target predictor or none. And measurement levels such as

continuous nominal or flag for all variables. The term flag is used to

denote a variable with two categories one of which can be considered positive

and the other negative. In this example the measurement level for the churn

field is set to flag and the role is set to target. All others are set as

predictors and inputs. The original data set has many fields and some of them

are not relevant to the target variable, so we first need to decide which fields

are more useful as predictors. There is a feature selection modeling node that

helps to do this. After the stream with the feature selection node is executed a

yellow model nugget gets created below it in the flow diagram.Using that nugget

we can generate a filter node that filters out the variables that are not

good predictors for the target. The data audit node located below the filtering

node shows various properties of the data such as numbers of outliers in each

variable and the percentage of valid values. It can also help to create a

special node for missing value imputation that is replacing missing

values of a variable with some valid values that can be selected based on

domain knowledge. Here variable log toll has greater than 50% missing values and

we will specify a value the mean to replace them. A super node in modeler is

a special node that is not found in the palette but is created by the user with

special functions included in it. The data audit node enables us to create a

super node for imputing missing values. It is shaped as a star and shown on the

right of the screen. Finally we attach the logistic

regression model node to the stream and click run. Another model nugget appears

and by clicking it we can see various model information and other output. in the output window that opens when we click on the model nugget the summary

tab shows the target inputs and some model building settings. Based on certain

advanced output settings that were specified before the model was built we

can also see a classification table, accuracy, and some other generated

outputs for the model. Note that these results are based on training data only.

To assess how well the model generates two other real-world data you should

always use a partition node to hold out a subset of records for the purposes of

testing and validation. Then, in the model setup screen select the use partitioned

data check box. This will help detect and avoid model

overfitting. Overfitting is defined as having significantly higher accuracy on

the training data. Data used for training the model then on tests or unseen data.

The yellow model nugget added earlier can also be used to compute predictions,

also called scores on the original data or on a new data source. All we need to

do is to connect the data source in question to the nugget, make sure it has

the predictor variables used in the model, and create an output to a table or

other structure for storing the scores. We can also specify settings for scoring

inside the model nugget. Note that if the model was built on transformed predictor

data, the same data transformation steps would be applied to the new data before

it can be scored by the model. The analysis node is the final node in the

stream. It attaches to a model nugget and when executed it will compute some model

evaluation metrics, auch as a confusion matrix and accuracy. In this example

we've only looked at a logistic regression model. IBM SPSS Modeler offers

a rich modeling palette that includes many classification, regression

clustering, Association rules and other models. It also contains large selections

of data source types, data transformations, graphs,

and output notes. And we haven't even talked about text analytics, entity

resolution and many other features of the product that can be extremely

helpful to data scientists. We could create an entire course on IBM SPSS

Modeler alone. You've learned how IBM SPSS Modeler helps analysts to create

powerful machine learning pipelines using graphical interface. Next, we will

talk about the original SPSS product now called IBM SPSS Statistics.

**SPSS Statistics**

1. 1968: Called Statistical package for social sciences
2. Statistical and machine learning software application
   1. Widely used in academia, government agencies and large enterprises
3. Used to build predictive models, perform statistical analysis of data, and conuct other analytical tasks
4. Visual interface

IBM SPSS Statistics evolved from an original product that was released in 1968. That product

was called “Statistical Package for Social Sciences,” or “SPSS.”

IBM SPSS Statistics is a statistical and machine learning software application and is widely

used in academia, government agencies, and large enterprises. It’s used to build predictive

models, perform statistical analysis of data, and conduct other analytic tasks. It has a

visual interface, which enables users to leverage statistical and data mining algorithms without

programming, although the interface is very different from Modeler. As you can see, the

main section of the screen looks very much like a spreadsheet; it displays data and allows

manual editing. This particular small data set, called “Employee Data”, was created

some time ago and does not represent real people. It is shipped with the product for

use in demos and tutorials.

At the bottom of the screen, we can see two tabs: Data View and Variable View. In the

Variable View, we can see and edit the information about all variables, including names, labels,

data types, and measurement levels. We can also specify labels for values of categorical

variables, and missing values.

At the top of the data window is a menu. Under File, if you select “Import Data,” you

will see a list of a wide variety of data formats that you can import. The product uses

its own data file format with the extension “.sav” that saves all the information

about the variables we just saw in Variable view. The menu enables importing from and

exporting to many other formats.

Under “Data,” you’ll find an extensive menu of possible data operations. Note that

Data Validation can be performed using user-defined rules that specify the expected behavior of

variable values. For example, if the date and month are kept in separate columns, the

date cannot exceed “31,” but for February, the date can’t exceed “29.” A special

rule can therefore be created and applied during data validation. Additionally, you

can enable some checks, such as percentage of missing values in a record or in the field.

When you click the “Transform” menu item, you’ll find a variety of available data

transformations. Under “Compute Variable…” you can write

a formula for a new variable based on existing variables. You can use any of the many mathematical

and statistical functions available in the product.

You also have the option to use automatic data preparation, similar to Modeler.

In the “Analyze” menu, you will see many types of statistical and machine learning

analysis. Under “Regression,” there are a variety of regression-related models. There

are other kinds of regressions that appear separately on the Analyze menu, including

General Linear Model, Generalized Linear Models, Mixed Models, and Loglinear.

Now let’s build a decision-tree model on the data. For this exercise we’ll try to

predict the "Employment category" field based on other fields. In the “Analyze” menu,

select “Classify” and then “Tree”. <Click> In the Decision Tree window, we can

specify the dependent variable “Employment Category,” and use most other fields -- except

id and bdate -- as predictors, or independent variables. Usually the ID variable should

not be used as a predictor, because it will not help with new cases, and the birthdate

does not seem to be a useful predictor in this example either. We’ll select “Exhaustive

CHAID” as our Growing Method, although there are also three other options available. Data

scientists often try many different models to see which one works best for their data.

Here we are just looking at one example model in order to illustrate how the product works.

Click the “Validation” button to open the Decision Tree Validation window. Here,

we select “Split-sample validation” to make sure we test the model on new data. Click

“OK” in the Decision Tree window, to <Click> generate the output, including the tree diagram

shown here. <Click> A Classification table is also displayed that shows how well the

model works on training and test data. In this case, the accuracy is 91.2% on training

data and only 85.6% on test data, which means the model does not generalize to new data

very well. It’s possible that by using different models, we can get better results.

Let’s move to the next menu item. When you click “Graphs,” you’ll open a versatile

Chart Builder, in addition to several other options.

The Chart Builder enables us to choose a style from the gallery and to drag required fields

onto the canvas, select colors, and choose from other options.

Here’s an example after we drag the “Previous Experience,” “Current Salary,” and Gender

variables to the corresponding slots to define the axis and colors for the dots on the chart.

The plot in the canvas is not based on real data, this example simply gives you an idea

of what to expect.

Here is the real plot obtained from the data that we’ve been using. It shows different

colored dots for gender, and regression lines that show the relationship of the current

salary to previous experience for each gender.

Throughout IBM SPSS Statistics, you’ll see a “Paste” button. When you click the “Paste”

button, instead of executing the task right away the application will open another window,

called the Syntax editor. Here, you can see the code called “syntax” pasted for you.

SPSS syntax is a special programming language.

For example, here is the code for the decision tree we just built. Once we have the syntax,

we can execute it, manually edit it, store it for later use, or send it to other users

of IBM SPSS Statistics. Experienced SPSS users can write the code from scratch, while others

might prefer to have it generated by the graphical interface. Remember, the option to paste syntax

is available in throughout the program. If the syntax is generated by all the steps

in a data analytics process -- opening the data set, applying any data transformations,

building models -- and then saved as a syntax file with the extension “.sps”, it’s

similar to saving a stream in IBM SPSS Modeler. However, one important difference is that

it does not allow for an easy way of scoring new records with the model. We’ll talk about

different ways to deploy models in the next section.

You’ve learned how IBM SPSS Statistics helps data scientists to analyze their data using

many statistical and machine learning techniques. Using a graphical user interface, we can create

complicated analysis that can be saved in the form of syntax and reused later.

Next, we will talk about predictive model deployment, an important part of the overall

data science lifecycle.

**Model deployment with Watson Machine Learning**

1. How to deploy model
   1. Workflow solutions (tend to support only models built with the same product)
   2. Open standards
2. Solutions to solve this problem
   1. Sage Maker by Amazon
   2. MLFlow by Databricks
   3. Airflow from Airbnb, now at Apache
   4. Kubeflow from Google
3. PMML (predictive model markup language)
   1. 1990s by data mining group
   2. XML based based, version 4.4 has 17 models, many transformations and model combination mehods
   3. Supported by 30 companies
   4. Can be generated by Watson Studio
4. PFA. Portable Format for analytics from DMG
   1. JSON based small programming language
   2. 5-6 years old
5. Supported by several companies
6. ONNX: Open Neural Network Exchange
   1. Supports all machine learning
7. IBM Watson Machine Learning
   1. Support for PMML and ONNX
   2. Deployment can be done using graphical interfae or python code, and can be for online scoring through a REST API or batch scoring

So far, we’ve talked about building machine learning models and pipelines. In most practical

applications, the return on investment is obtained when the model or pipeline is put

into production, where it is used to get predictions, or scores, for the new cases.

Let’s look back at our overview of different tool categories. In this unit, Model Deployment

is our focus.

Suppose you worked hard to create the best possible machine learning model and the data

preparation pipeline for it. How will you deploy your models?

In many practical scenarios, models are built and deployed by different teams, using different

programming, and perhaps human languages. The teams will use different computing and

data storage environments, and It might prove difficult to translate your program and the

associated data preparation and post-processing steps from one environment to the other.

Currently there are several approaches you can use to solve this problem, some commercial,

some open source. Yet each one typically supports only a subset of all possible models, from

building them to deploying, so a user gets locked into a specific framework.

Open standards for model deployment are designed to support model exchange between a wider

variety of proprietary and open source models. Predictive Model Markup Language, or “PMML,”

was the first such standard, based on XML.

It was created in the 1990s by the Data Mining Group, a group of companies working together

on the open standards for predictive model deployment. IBM and SPSS were among the founding

members of the Data Mining Group.

PMML 4.4 was recently released.

It includes 17 statistical and machine learning models and many data transformations, built-in

functions, ways to combine multiple models together, and other features. This standard

is widely known and used. The products we looked at earlier -- Watson Studio, IBM SPSS

Statistics, IBM SPSS Modeler -- enable users to export most models in PMML.

In 2013, a demand for a new standard grew, one that did not describe models and their

features, but rather the scoring procedure directly, and one that was based on JSON rather

than XML. This led to the creation of Portable Format for Analytics, or PFA. PFA is now used

by a number of companies and open source packages.

After 2012, deep learning models became widely popular. Yet PMML and PFA did not react quickly

enough to their proliferation. The need for a standard intermediate representation was

amplified by the wide variety of emerging deep learning frameworks and specialized hardware.

In 2017, Microsoft and Facebook created and open-sourced Open Neural Network Exchange

, or “ONNX.” Originally created for neural networks, this format was later extended to

support “traditional machine learning” as well.

There are currently many companies working together to further develop and expand ONNX,

and a wide range of products and open source packages are adding support for it.

Watson Machine Learning is IBM’s commercial offering designed for model deployment. It

supports deployment of models built with most open source packages, as well as those expressed

in PMML or ONNX. It also supports deployment of IBM SPSS Modeler streams and Modeler flows

from Watson Studio. Deployment can be done using a graphical interface or Python code,

and can be for online scoring through a REST API or batch scoring.

Watson Machine Learning helps integrate a deployed model into applications in the form

of code snippets in several programming languages.

In this video, you’ve learned how open standards and Watson Machine Learning can help users

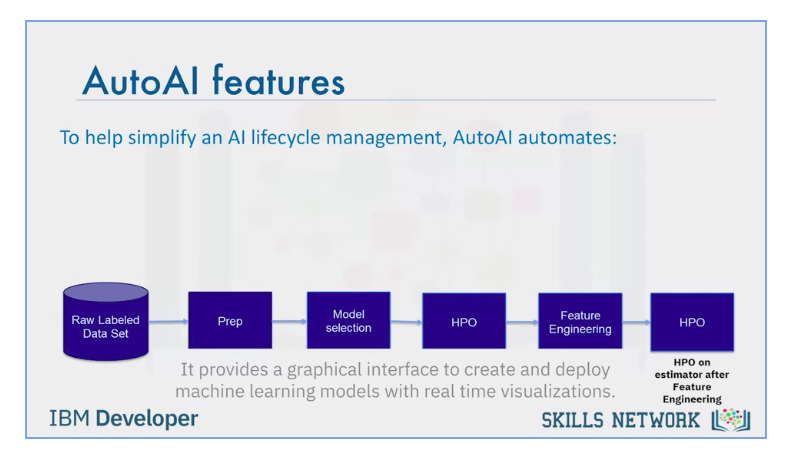
to deploy their models into various application.

Next we’ll talk about AutoAI and OpenScale, two advanced Watson Studio features that help

to further simplify a data scientist’s work.

**Auto AI in Watson studio**

1. Graphically create a stream or flow that includes dat transofrmaiton steps
   1. These are called data pielines or ml pipelines



1. Auto AI only for classification and regression

In earlier sections we saw how IBM SPSS Modeler and Watson Studio Modeler flows allow you

to graphically create a stream or flow that includes data transformation steps and machine

learning models.

Such sequences of steps are called data pipelines or ML pipelines.

This section examines a feature of Watson Studio that helps to automate the creation

of machine learning pipelines.

This allows data scientists to produce results much faster and to focus on more creative

work.

There is currently a shortage of qualified data scientists.

Many operations that a data scientist typically performs are repetitive and time-consuming.

Therefore, automating some of that repetitive work will help free up both new and experienced

data scientists to do the important work that they are trained to do.

The AutoAI system was developed by IBM Research experts in collaboration with IBM Distinguished

Engineer and two-time Kaggle Grandmaster Jean-Francois Puget.

It provides a graphical interface to create and deploy machine learning models with real

time visualizations.

AutoAI automatically performs typical machine learning steps, such as:

Data preparation Model selection

Feature engineering Hyper-parameter optimization

Users can view the progress on the graphical interface.

This example shows the training of a model to predict whether or not a customer is likely

to buy a tent from an outdoor equipment store.

We start with structured data.

In this historical data, there are four feature, or “predictor,” columns:

GENDER: The customer’s gender AGE: The customer’s age

MARITAL\_STATUS: “Married”, “Single”, or “Unspecified”

and PROFESSION: The general category of the customer’s

profession, such “Hospitality” or “Sales”, or simply “Other.”

The model will learn to predict the value for the IS\_TENT column; that is, whether or

not the customer bought a tent.

After we choose IS\_TENT as the column to predict, AutoAI analyzes the data and determines that

the IS\_TENT column contains True/False information, making this data suitable for a binary classification

model.

The default metric for a binary classification is ROC/AUC.

After we click Run experiment, an infographic shows the process of building the pipelines

as the model trains.

Once the pipeline creation is complete, we can view and compare the ranked pipelines

in a leaderboard.

The pipelines for the sample binary classification model are quite uniform because of the underlying

sample data.

To see pipelines in action, re-run the experiment as a regression experiment to predict purchase

amount.

That experiment gives better variation in the resulting pipelines.

After clicking “Pipeline comparison,” we can see how the pipelines differ on various

measures of model quality.

The pipelines can be saved as Machine Learning assets in the Watson Studio project.

Then they can be deployed and tested.

Currently AutoAI is available only for classification and regression models; there is a plan to

add time series model support in the future.

In this unit, you have learned how AutoAI automates typical data science tasks and helps

get better performing data pipelines more quickly, while also simplifying pipeline deployment

into production in Watson Machine Learning.

In the next section, we will discuss Watson OpenScale, which helps to ensure that your

models are fair, explainable, and up to date.

**Watson Open Scale**

1. Fairness: Detect and mitigate model bias; automatically recommend which attributes to monitor for bias
2. Explainability: Audit and explain model decisions
3. Model Monitoring: Monitor model performance, help to find causes when odel drift is detected, possibly trigger retraining
4. Business Imoact: Correlate model metrics and business KPIs to measure business impact; actionable meterics and alerts
5. Explanatons provided by Watson open scale enhances compliance with regulations such as
   1. Fair credit reporting Act
   2. GDPR
6. Customers the rights to add why their applications were denied
7. Drift: Less accurate data
   1. When it hits a chosen threshold it creates an alert and explains what trnsactions caused the rift
   2. Can be used to retrain the model so its predictive accuracy does not dro
   3. Highlights route causes
8. Fair, explainable and compliant