Course 2

Tools for Data Scientists

Contents

[Data Science Task Categories 5](#_Toc143011742)

[1. Data Management: 6](#_Toc143011743)

[2. Data Integration and Transformation: 8](#_Toc143011744)

[3. Data Visualization: 9](#_Toc143011745)

[4. Model Building: 12](#_Toc143011746)

[5. Model Deployment: 13](#_Toc143011747)

[6. Model monitoring and Assessment: 14](#_Toc143011748)

[7. Code Asset Management: 15](#_Toc143011749)

[8. Data Asset Management (Data Governance/ Data Lineage): 15](#_Toc143011750)

[9. Development Environment: 16](#_Toc143011751)

[10. Execution Environment: 17](#_Toc143011752)

[11. Fully integrated Visual Tools: 18](#_Toc143011753)

[Introduction to Python 20](#_Toc143011754)

[Who are Python Users 20](#_Toc143011755)

[Benefits of Using Python 21](#_Toc143011756)

[Diversity and Inclusion Efforts of Python Community 21](#_Toc143011757)

[Introduction to R Programming Language 22](#_Toc143011758)

[Open-Source Vs Free Software 22](#_Toc143011759)

[Who Use R Language 22](#_Toc143011760)

[Benefits of Using R 22](#_Toc143011761)

[Global R Communities 23](#_Toc143011762)

[Introduction to SQL 23](#_Toc143011763)

[SQL and Relational Databases 24](#_Toc143011764)

[SQL Elements 24](#_Toc143011765)

[Benefits of Using SQL 25](#_Toc143011766)

[Other Languages for Data Science 25](#_Toc143011767)

[Review languages & Their Relationship with Data Science 26](#_Toc143011768)

[Java 26](#_Toc143011769)

[Scala 26](#_Toc143011770)

[C++ 26](#_Toc143011771)

[JavaScript 27](#_Toc143011772)

[Julia 27](#_Toc143011773)

[Scientific Computing Libraries 29](#_Toc143011774)

[Visualization Libraries 29](#_Toc143011775)

[High-Level- Machine Learning and Deep Learning Libraries In Python 30](#_Toc143011776)

[Deep Learning Libraries in Python 30](#_Toc143011777)

[Libraries in Other Languages 30](#_Toc143011778)

[Summary: 31](#_Toc143011779)

[API Definition 32](#_Toc143011780)

[List of API Libraries 32](#_Toc143011781)

[REST Libraries 32](#_Toc143011782)

[Summary: 33](#_Toc143011783)

[Definition of Data Set 34](#_Toc143011784)

[Types of data ownership 34](#_Toc143011785)

[Sources of data 35](#_Toc143011786)

[Community Data License Agreement 35](#_Toc143011787)

[Open datasets and sources 36](#_Toc143011788)

[Government Data: 36](#_Toc143011789)

[Financial Data Sources: 36](#_Toc143011790)

[Crime Data: 36](#_Toc143011791)

[Health Data: 36](#_Toc143011792)

[Academic and Business Data: 37](#_Toc143011793)

[Other General Data: 37](#_Toc143011794)

[Propriety datasets and sources 37](#_Toc143011795)

[Dataset licenses 37](#_Toc143011796)

[1. Public domain mark - PUBLIC DOMAIN 37](#_Toc143011797)

[2. Open data commons public domain dedication and license – PDDL 37](#_Toc143011798)

[3. Creative commons attribution 4.0 international CC-BY 38](#_Toc143011799)

[4. Community data license agreement – CDLA PERMISSIVE-2.0 38](#_Toc143011800)

[5. Open data commons attribution license - ODC-BY 38](#_Toc143011801)

[6. Creative commons attribution-sharealike 4.0 international - CC-BY-SA 38](#_Toc143011802)

[7. Community data license agreement – CDLA-SHARING-1.0 38](#_Toc143011803)

[8. Open data commons open database license - ODC-ODBL 38](#_Toc143011804)

[9. Creative commons attribution-noncommercial 4.0 international - CC BY-NC 38](#_Toc143011805)

[10. Creative commons attribution-no derivatives 4.0 international - CC BY-ND 38](#_Toc143011806)

[11. Creative commons attribution-noncommercial-sharealike 4.0 international - CC BY-NC-SA 38](#_Toc143011807)

[12. Creative commons attribution-noncommercial-noderivatives 4.0 international - CC BY-NC-ND 38](#_Toc143011808)

[Summary 41](#_Toc143011809)

[Machine Learning 42](#_Toc143011810)

[Supervised Learning 42](#_Toc143011811)

[Regression Models 43](#_Toc143011812)

[Classification Models 43](#_Toc143011813)

[Unsupervised Learning: 43](#_Toc143011814)

[Reinforcement Learning: 43](#_Toc143011815)

[Deep Learning: 43](#_Toc143011816)

[Summary 44](#_Toc143011817)

[MAX Reduces Time to Value 45](#_Toc143011818)

[Typical model-serving microservice 45](#_Toc143011819)

[Summary 46](#_Toc143011820)

[Jupyter Notebook 48](#_Toc143011821)

[Jupyter Lab 48](#_Toc143011822)

[Summary 49](#_Toc143011823)

[Hands On to Jupyter Notebook 50](#_Toc143011824)

[Some Shortcuts for Working in Jupyter Notebook 51](#_Toc143011825)

[Summary 51](#_Toc143011826)

[Introduction to Kernels 52](#_Toc143011827)

[Summary 52](#_Toc143011828)

[Client and Kernel 53](#_Toc143011829)

[Saving a Jupyter Notebook 53](#_Toc143011830)

[Summary 53](#_Toc143011831)

[Computational Notebooks 54](#_Toc143011832)

[Jupyter Lab 54](#_Toc143011833)

[Anaconda 54](#_Toc143011834)

[VisualStudio Code 55](#_Toc143011835)

[Summary 55](#_Toc143011836)

[Computational notebooks 56](#_Toc143011837)

[JupyterLite 56](#_Toc143011838)

[Google Colaboratory 57](#_Toc143011839)

[Summary 57](#_Toc143011840)

[What is R? 58](#_Toc143011841)

[RStudio 59](#_Toc143011842)

[Tabs in RStudio 59](#_Toc143011843)

[Popular R Libraries for Data Science 59](#_Toc143011844)

[Summary 59](#_Toc143011845)

Module 1

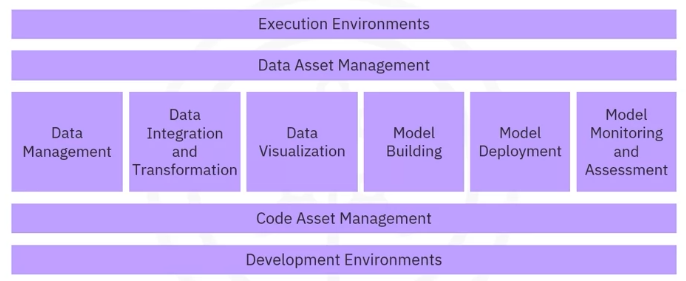
Tools For Data Science

# Data Science Task Categories

1. Data Management
2. Data integration and transformation
3. Data visualization
4. Model building
5. Model deployments
6. Model monitoring and assessment

The data scientists need following four things to achieve the above mentioned tasks:

1. Data asset management
2. Code asset management
3. Development environment
4. Execution environment



### Data Management:

* It involves collecting, persisting and retrieving data securely, efficiently and cost effectively.
* The sources of the data could be twitter, media, internet, sensors and Flipkart.
* Commercial supports delivered by software vendors, influential partners and support networks.
* In data management, with some exceptions, software-as-a-service (SaaS) versions of existing open source and commercial tools exist. The cloud provider operates the tool for you in the cloud. For example, the cloud provider operates the product by backing up your data and configuring and installing updates.

#### Tools for Data Management:

The relational databases include:

|  |  |  |
| --- | --- | --- |
| Open-Source Tools | Commercial Tools | Clouds |
| MySQL  PostgreSQL | **Oracle database** | Some proprietary tools are only available from a single cloud provider.  **DynamoDB:** A noSQL database that is a Web Service from Amazone. It allows storage and retrieving data in a key-value or a document store format. The most prominent document data structure is *JSON*. |
| NoSQL Databases include:  Mongo DB,  Apache couch DB,  Apache Cassandra | **Microsoft SQL Server** | **Cloudant:** A database as a service offering. But, in the background, it is based on the open-source *Apache CouchDB*. The advantage is that complex operational tasks like updating, backup, restoring and scaling are done by the cloud provider. However, the Cloudant service offering is compatible with CouchDB. Therefore, the application migrates to another CouchDB server without making any changes to the application. |
| File based tools like Hadoop file system, cloud file system like Ceph | **IBM BD2** | **IBM DB2:** A commercial database made available as a SaaS offering in the cloud, taking away operational tasks from the user. |
| Elastic search tool (Stores text data with search index) |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

In summary:

* MySQL, and PostgreSQL are examples of Open Source Relational Database Management Systems (RDBMS), and IBM Db2 and SQL Server are examples of commercial RDBMSes and are also available as Cloud services.
* MongoDB and Apache Cassandra are examples of NoSQL databases.
* Apache Hadoop and Apache Spark are used for Big Data analytics.

### Data Integration and Transformation:

* Also termed Data Refinery and Cleansing, it deals with the extraction, transformation and loading (ETL) of data.
* Normally, the data is distributed in multiple repositories like database, data cube and flat files. This procedure tends to extract the data from all these repositories, transform it and then store it (loading) into a central repository that is termed as the data warehouse.
* The transformation involves changes into the values, structure and format of data. Once the data is transformed, it is loaded to the data warehouse.
* The transformation steps are pushed towards the domain of the data scientist or data engineer.

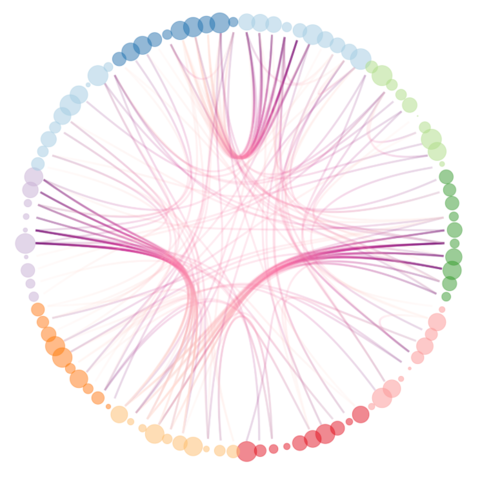
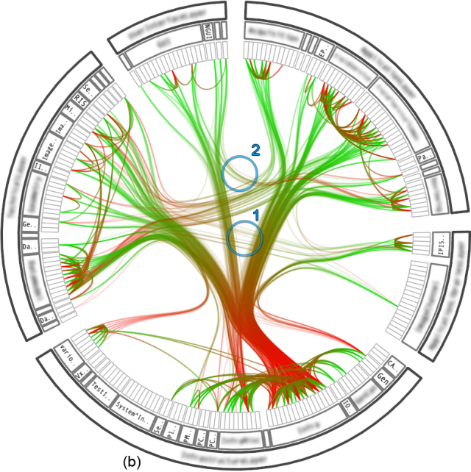
|  |  |  |
| --- | --- | --- |
| Open-Source Tools | Commercial Tools | Clouds |
| Apache airflow (created by airbnb) | **Informatica PowerCenter** | **Informatica Coud Data Integration** |
| Kubeflow that allows execution of data science pipelines on top of Kubernetes. | **IBM Infosphere DataStage** | **IBM Data Refinery:** It is a part of *IBM Watson Studio*.  It allows transforming large amounts of raw data into consumable, quality information in a spreadsheet-like user interface. |
| Apache Nifi: delivers nice visual editor | **SAP**  **Oracle**  **SAS**  **Microsoft Products**  **Talend:** Support the design and deployment of ETL data processing pipelines through a graphical interface.  Bring along connectors to most of the commercial and open-source target information systems. |  |
| Apache Spark SQL: allows use of ANSI SQL and scales up to compute clusters of thousands of nodes | **IBM Watson Studio Desktop:** Includes a component called Data Refinery, that enables definition and execution of data integration processes in a spreadsheet-style. |  |
| Node Red: Equipped with visual editor, it is so low in resource consumption that it runs on tiny devices, such as Raspberry Pi. |  |  |
|  |  |  |
|  |  |  |

### Data Visualization:

* It involves the graphical representation of data and information in the form of charts, plots, maps and animations.
* Various forms of presentation include bar charts, tree maps, line charts and map charts.
* These tools are supported by programming libraries where you need to use codes.
* In the commercial environment, data visualization use business intelligence (BI) tools with a prime focus on creating visual reports and live dashboards.

|  |  |  |
| --- | --- | --- |
| Open-Source Tools | Commercial Tools | Clouds |
| Pixie Dust: This tool comes with user interface for plotting in python. | **Tableau** | **Datameer:** |
| Hue: It creates visualization with SQL queries. | **Microsoft Power BI** | **IBM Cognos Analytics:** A business intelligence suite as a cloud solution |
| Kibana: It is a data exploration and visualization tool (Web application) that is limited to ELASTIC SEARCH (Data provider). | **IBM Cognos Analytics** | **IBM Data Refinery:** It also offers data exploration and visualization functionality in *Watson Studio*. |
| Apache Superset: It is a web application-based data exploration and visualization tool. | **IBM Watson Studio Desktop:** visualization can show relationships between different columns in a table. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

* There are many different ways to visualize data. For instance:
* one data visualization is hierarchical edge bundling that depics correlations and affiliations between entities.



* 3D bar chart to visualize a target value on the vertical dimension, which is dependent on two other values in the horizontal dimensions.

A graph of different colored squares

Description automatically generated

* A graph of a number of red rectangular bars

  Description automatically generated with medium confidenceA classic bar chart
* A 2D scatter plot with a heat map shows two dependent data fields on the y-axis with different colour intensities.

A graph of purple dots

Description automatically generated with medium confidence

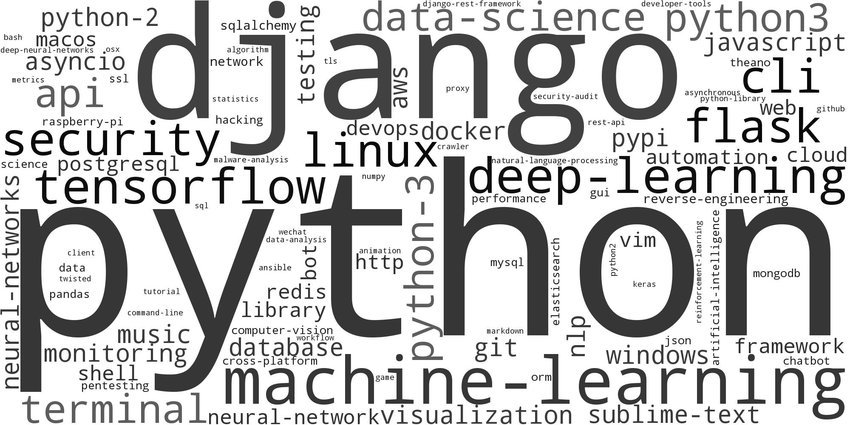
* A group of colored squares

  Description automatically generatedA tree map shows the distribution of subsets within a set
* A pie chart does the same, but in a non-hierarchical manner

A circular chart with different colored circles

Description automatically generated

* A word cloud pops out significant terms in a document corpus.



### Model Building:

* It includes the procedures to train data and analyze patters using suitable machine learning algorithms.
* By using a known data, a system is developed to provide predictions and decisions. The same system is then applied to a new unseen data to get predictions and decisions accordingly.
* Machine learning models can be built using IBM Watson Machine Learning.

|  |  |  |
| --- | --- | --- |
| Open-Source Tools | Commercial Tools | Clouds |
|  | **SPSS Modeler:** This modeler is also available in *Watson Studio Desktop* based on the tool’s cloud version. | **IBM Watson Machine Learning:** It can train and build models using various open-source libraries. |
|  | **SAS Enterprise Miner** | **Google AI Platform Training** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Model Deployment:

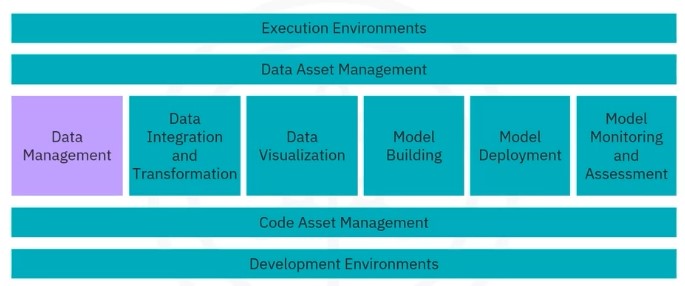
* It includes integrating a model into a production environment.
* By using APIs (application programing interface) to enable data-based decisions.
* For instance, SPSS collaboration and deployment services are used for model deployment.
* Model deployment in commercial software is tightly integrated into the model-building process.
* Commercial softwares can export models in an open format. For example, SPSS modeler supports exporting models as predictive model markup language (PMML) which an abundance of other commercial and open software packages can read.
* Model deployment in commercial software is usually tightly integrated into the model-building process.
* Commercial software can export models in an open format, such as PMML. For example, SPSS modeler supports exporting models as Predictive Model Markup Language (PMML) which other commercial and open software packages can read.

|  |  |  |
| --- | --- | --- |
| Open-Source Tools | Commercial Tools | Clouds |
| Apache Prediction IO: It only supports Apache Spark ML. | **SPSS Collaboration and Development Services:** Used to deploy any type of asset created by the SPSS software tools suite. | **SPSS Collaboration and Development Services:** |
| Seldon: It supports nearly every framework, for instance *Tensor Flow*, *Apache Spark ML*, *R*, *Scikit learn*. It can also run on top of Kubernetes and *Redhat Openshift*. |  | **IBM Watson Machine Learning:** It deploys a model and makes it available to consumers using a REST interface. |
| mleap: It supports Apache ML. |  | Amazon SageMaker Model Monitor: It is an example of a cloud tool to monitor deployed machine |
| Tensor Flow: It supports any Tensor Flow model. *Tensor Flow Lite* is designed for small devices such as Raspberry Pi whereas *Tensor Flow dot JS* is developed for web browsing. |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Model monitoring and Assessment:

* This task tracks deployed models either by using tools like Fiddler in production environment, or by using evaluation metrics such as F1 score, True positive rates and some of square error.
* IBM Watson open scale can be used for model monitoring and assessment.
* Currently, relevant commercial tools are not available and only open-source tools are being used for model monitoring and assessment.

|  |  |  |
| --- | --- | --- |
| Open-Source Tools | Commercial Tools | Clouds |
| Model DB: Manage ML models, information about models is stored and queried. Natively supports *Apache Spark MK* pipelines and Scikit learn. |  | **Amazone SageMaker Model Monitor:** It is a cloud tool to monitor deployed machine learning and deep learning models continuously. |
| Prometheus: It is not specifically developed for ML models; however, it is frequently used for this purpose. Model performance is measured more than accuracy. |  | **Watson OpenScale:** Everything marked green in the fibure after this table can be done by using Watson OpenScale. |
| IBM AI Fairness 360 Open-Source Toolkit: Model bias against protected groups such as gender or race is a problem in many different scenarios. This tool detects and mitigates such bias in ML models. |  |  |
| Adversarial Robustness 360 Toolbox: ML like artificial neuronal networks based deep learning models can be subjected to adversarial attacks. This tool kit detects vulnerability against adversarial attacks. This toolkit leverages models to be more robust. |  |  |
| IBM AI Explain ability 360 Toolkit: ML Models often considered black box applying some magic. This toolkit provides a solution to this problem by finding similar examples in a data set to be presented to an end-user for manual comparison. This toolkit is good for training a simpler ML model to explain responsibility of different input variables directed towards final decision of model. |  |  |
|  |  |  |
|  |  |  |



Operations that can be achieved using Watson OpenShift tools.

### Code Asset Management:

* This management system is used to fix bugs that is required for version control. It is also termed as version management or version control.
* For example, GitHub is used as code asset management.
* Commercial tools are not available for code assessment management. Therefore the open source tools are the first choice. For instance, Git and GitHub is the de facto standard.

|  |  |  |
| --- | --- | --- |
| Open-Source Tools | Commercial Tools | Clouds |
| Git |  |  |
| GitHub |  |  |
| GitLab: it is the toolkit with advantage of being entirely open source, can be hosted and managed on your own. |  |  |
| BitBucket |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Data Asset Management (Data Governance/ Data Lineage):

* Data must be versioned and annotated with metadata.
* It is a crucial part of enterprise-grade data science.
* These tools provide a platform for organizing and managing the data.
* For instance, the tools help the data scientists in replicating the data, in storing a backup of the data and in data success management (to protect it by a password in order to define the users who can access the data).
* Functions include:

1. Data governance
2. Data versioned and annotated
3. Data dictionary
4. Data lineage
5. Data privacy and retention

|  |  |  |
| --- | --- | --- |
| Open-Source Tools | Commercial Tools | Clouds |
| Apache Atlas | **Informatica Enterprise Data Governance** |  |
| Odpi Egeria (Linux based) | **IBM Infoshpere Information Governance Catalog:** Covers functions like a data dictionary, which facilitates the discovery of data assets. Each data asset is assigned to a data steward or the data owner. The data owner is responsible for that data asset and can be contacted. |  |
| Kylo: an open-source data management software platform. |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Development Environment:

* It provides an integrated development environment (IDEs).
* These tools serve as a workspace to work on source code in developing, implementing, executing, testing and deployment of source code.
* For instance, IBM Watson provides IDEs mentioned above.

|  |  |  |
| --- | --- | --- |
| Open-Source Tools | Commercial Tools | Clouds |
| Jupyter: Tools for interactive python programming. It supports more than 100 programming languages through “Kernels”. It encapsulates the execution environment for different programming languages. The best part is the unified documentation, since the code output from code, shell commands and visualization, all are seen on a single canvas. | IBM Watson Studio Desktop: most people consume it through the cloud. However, a desktop version is also available. It combines Jupyter Notebooks with graphical tools to maximize the performance. |  |
| Jupyter Lab: It is more modern and modular. It is the next version of jupyter notebooks and will replace the notebooks in future. These tools can open jupyter notebooks, data terminals and then arrange them on canvas. |  |  |
| Apache Zeppelin: It is inspired by jupyter notebooks and provides similar experience. In jupyter notebook, you are required to use external libraries in Zeppelin and plotting doesn’t require coding. |  |  |
| R Studio: It exclusively runs R and all associated R libraries. Python development is possible in R environment. R is tightly integrated into Jupyter tool and provides optimal user experience. It unifies programming, execution, debugging, remote data access, data exploration and visualization into one tool. |  |  |
| Spyder: It mimic behavior of R Studio to bring its functionality to python world. Data scientists consider it as alternative of Jupyter. In python world, Jupyter is used more. Spyder integrates codes, documentation and visualization into a single canvas. |  |  |
|  |  |  |
|  |  |  |

### Execution Environment:

* These tools provide cluster execution environment with libraries for code compiling and system resources to execute and verify codes.
* Cloud based execution environments are not tied to specific hardware and software. For instance, IBM Watson.

|  |  |  |
| --- | --- | --- |
| Open-Source Tools | Commercial Tools | Clouds |
| Apache Spark: provides linear scalability i.e., double the clusters, double is the performance. |  |  |
| Apache Flink: There is a difference between Apache Spark and Flink. Spark is a batch data processing engine, capable of processing vast amount of data one by one or file by file. However, Flink stream processing image and capable of processing real time data streams. |  |  |
| Riselab RAY: It is the latest development in data science with a clear focus on large-scale deep learning model training. |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Fully integrated Visual Tools:

* These include IBM Watson Studio and IBM Cognos Dashboard Embedded.
* To use these tools, there is no compulsion of having a previous programming experience.
* These tools provide data integration and transformation, visualization and model building.

|  |  |  |
| --- | --- | --- |
| Open-Source Tools | Commercial Tools | Clouds |
| KNIME: originated from the University of Konstanz in 2004, it has drag and drop capabilities. These tools provide built in visualization capabilities and can be extended by programming in R and Python. It may also be linked with Apache Spark with connectors. | **Watson Studio** together with **Watson Open Scale** is a fully integrated tool covering the data science life cycle involving all tasks discussed previously.  They can be deployed in a local data center, on top of *Kubernetes*/ *RedHat OpenShift*. | **Watson Studio**  **Watson OpenScale**  **Microsoft Azure Machine Learning**  These three platforms cover the complete development life cycle for all data science, machine learning and AI. |
| Orange: Less flexible than KNIME, but easier to use. | **H2O Driverless AI:** which covers the complete data science life cycle. | **H2O Driverless AI:**  Since the cloud provider does not do operations and maintenance, as with Watson Studio, OpenShift and Azure, this delivery model should be distinct from Platform or Software as a service – SaaS or PaaS (Plateform as a service). |

Module 2

Programming Languages in Data Science

* Different programming languages have their own strengths and weaknesses.
* The choice largely depends on the type of data and the task that a data scientist has to work on.
* The most popular programming languages in data science (DS) are:
* Python
* R
* SQL (oldest programming language)
* The data scientists also use other languages like:
* Scala (from “scalable” + “language”)
* Java
* C++
* Julia (newest programming language)
* Some other languages have their use in unique cases, such as
* Java Script
* php
* Go
* Rubby
* Visual Basic
* Decision depends on:
* The thing you need to accomplish.
* The problem you need to solve.
* The company you work for
* Your role in the company
* Age of your existing application
* A data scientist may have different kind of roles in the industry such as:
* Business Analyst
* Database Engineer
* Data Analyst
* Data Engineer
* Data Scientist
* Research Scientist
* Software Engineer
* Statistician
* Product Manager
* Project Manager

# Introduction to Python

* Python is the powerhouse of a language.
* Most widely used and popular programming language in DS
* As per 2019 Kaggle Data Science and Machine Learning Survey, ¾ of the over 10,000 respondents worldwide reported that they use python regularly.
* According to Glassdoor in 2019, more than 75% of positions listed in DS included Python in their job description.
* Most of the aspiring data scientists opted Python to start with.

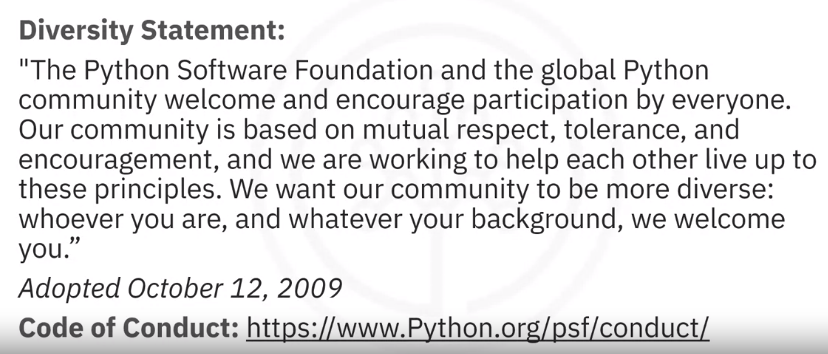
## Who are Python Users

* People who already know how to program. Because it uses clear and readable syntax. You can develop the same programs from other languages with lesser code using Python.
* For beginners, because of the huge global community and wealth of documentation. According to many different surveys in 2019, more than 80% of data professionals use Python worldwide.
* It is a useful language in areas like
* DS
* AI
* Machine Learning
* Web development
* Internet of Things (IoT) with devices like Raspberry Pi
* Large organizations that heavily use Python include:
* IBM
* Wikipedia
* Google
* Yahoo!
* CERN
* NASA
* Facebook
* Amazone
* Instagram
* Spotify
* Reddit
* Python is widely supported by a global community and shepherded by the Python Software Foundation.

## Benefits of Using Python

* It is a general-purpose language.
* 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.
* For **DS**, it has scientific computing libraries like **Pandas**, **NumPy**, **SciPy** and **Matplotlib**.
* For **AI**, it has libraries like **TensorFlow**, **PyTorch**, **keras**, and **Scikit-learn**.
* Can be used for ***natural language processing*** (NLP) using the natural language toolkit (NLTK)

## Diversity and Inclusion Efforts of Python Community

* Python community has a well-documented history of paving the way for diversity and inclusion efforts in the tech industry as a whole.
* Python community “***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 Programming Language

* According to the 2019 Kaggle Data Science Survey, over 10,000 respondents workwise, learning three languages can earn you an increment! With R as one of the three choices.

## Open-Source Vs Free Software

* Python is an Open-Source whereas R is a free software.

|  |  |
| --- | --- |
| Similarities | Differences |
| Both are free to use | The Open-Source Initiative (OSI) champions open source while the Free Software Foundation (FSF) defines free software |
| Both commonly refer to the same set of licenses, for example, many open-source projects use the General Public License (GNU) | Open-Source is more business focused while free software is more focused on a set of values |
| Both support collaboration |  |
| In many cases, these terms can be used interchangeable (but not all) |  |

* You should learn R because it is free software.
* You can use the language in the same way that you contribute to open source.
* In addition, it allows for private use, commercial use, and public collaboration.
* It is supported by a wide global community of people who want to use the language to solve big problems.

## Who Use R Language

* Statisticians, mathematicians, and data miners for developing statistical software, graphing, and data analysis.
* Someone with no or minimal programming background. R language’s ***array-oriented syntax*** makes it easier to translate from math to code for learners with no or minimal programming background
* For learners with a DS career
* R is popular in academia
* Companies like IBM, Google, Facebook, Microsoft, Bank of America, Ford, TechCrunch, Uber and Trulia.

## Benefits of Using R

* It is the largest repository of statistical knowledge.
* It has more than 15,000 publicly released packages to conduct complex exploratory data analysis.
* It integrates well with other computer languages like C++, Java, C, .Net and Python.
* Common mathematical operations like matrix multiplication give immediate results.
* It has stronger object-oriented programming facilities than most statistical computing languages.

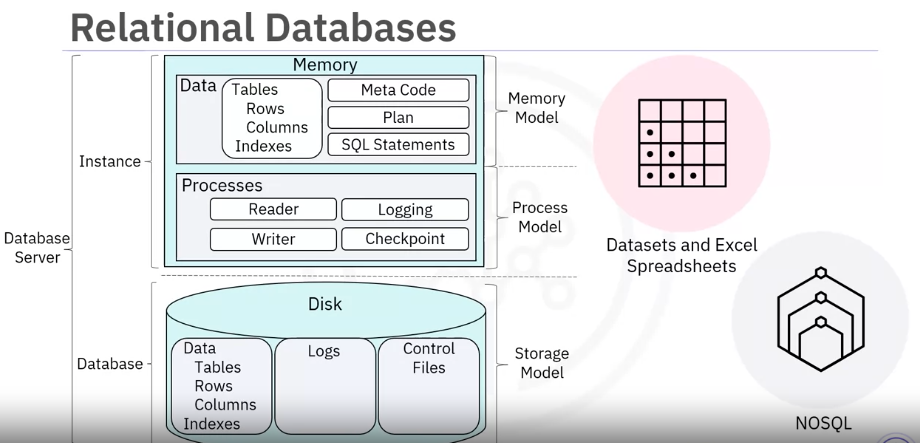
## Global R Communities

* To connect with other R users, you can join communities like:
* UseR!
* WhyR?
* SatRdays
* R-ladies
* In addition, you can check out the R project website for R conferences and events.

# Introduction to SQL

* It is officially pronounced “ess cue el”, though some people say “sequel”.
* SQL stands for Structured Query Language
* Many people consider SQL different from the other software development languages because it is a non-procedural language.
* Its scope is limited to querying and managing data.
* While it is not a DS language, the data scientists use this language regularly because it is simple and powerful.
* It is older than Python and R by about 20 years.
* It first appeared in 1974 and was developed at IBM.
* It is useful in handling structured data which is the data incorporating relations among entities and variables.

## SQL and Relational Databases

* SQL was designed for managing data in relational databases.
* A **relational database** is formed by collections of two-dimensional tables, for example datasets and Excel spreadsheets.
* Each of these tables is then formed by a fixed number of columns and any possible number of rows.
* However, although SQL was originally developed for use with relational databases, because of its pervasiveness and ease of use, SQL interfaces have also been developed for many NoSQL and bid data repositories.

## SQL Elements

* The SQL language is subdivided into several languages’ elements, including:
* Clauses
* Expressions
* Predicates
* Queries
* A diagram of a country

  Description automatically generatedStatements

## Benefits of Using SQL

* Knowing SQL will help you get many different jobs in DS and data engineering, such as a business and a data analyst.
* When performing operations with SQL, the data is accessed directly, without needing to copy the data separately, which can considerably speed up workflow executions.
* SQL behaves like an interpreter between you and the database.
* SQL is an American National Standards Institute (or ANSI), which means if you learn SQL and use it with one database, you can apply your SQL knowledge to many other databases easily.
* Many SQL databases are available, including:
* MySQL
* IBM DB2
* PostgreSQL
* Apache Open Office Base
* SQLite
* Oracle
* MariaDB
* Microsoft SQL Server
* The syntax of the SQL you write may change based on the relational database management system you are using.
* If you want to learn SQL, you should focus on a specific relational database and then plug into the community for that specific platform.
* In addition, there are many available great introductory courses on SQL.

# Other Languages for Data Science

* Previously, we reviewed Python, R and SQL.
* However, some other programming languages such as Scala, Java, C++, and Julia are probably the most traditional data science languages.
* Further, Java script, php, Go, Rubby and Visual Basic and many others have found their places in the data science community.

## Review languages & Their Relationship with Data Science

### Java

* A general-purpose object-oriented programming language
* It has huge adoption in the enterprise space, designed to be fast and scalable.
* Java applications are compiled to ***bytecode*** and run on the ***Java Virtual Machine*** or JVM.
* For DS, Java tools are:
* **Weka** (data mining)
* Hava-ML (Machine Learning Library)
* Apache MLlib (Scalable machine learning)
* Deeplearning4 for deep learning
* **Hadoop** is another application of Java which manages data processing and storage for big data applications running in clustered systems.

### Scala

* It is general-purpose programming language that provides support for functional programming and is a strong static type system.
* The Scala language was constructed to address the shortcomings of Java.
* Designed as an extension to Java, it is inter-operatable with Java as it also runs on JVM.
* The name Scala is a combination of scalable and language
* This language is designed to evolve with the requirements of its users.
* For DS, the most popular programs built with Scala include:
* **Apache Spark**:
* It is a fast and general-purpose cluster computing system that provides APIs, which make parallel jobs easy to write.
* It has an optimized engine that supports general computation graphs.
* Includes Shark (a query engine), MLlib (for machine learning), GraphX (for graph processing), and Spark Streaming
* It was designed to be faster than Hadoop.

### C++

* A general-purpose language, an extension of C programming language or “C with Classes”.
* It 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 rely on C++ to develop programs that feed data to customers in real-time.
* For DS, the most popular programs with C++ include:
* **TensorFLow**: a deep learning library for data flow. Although C++ is the foundation of TensorFlow, it runs on a python interface, so users don’t require the knowledge of C++ to run it.
* **MongoDB** is a NoSQL database for big data management that was built with C++.
* **Caffe** is a deep learning algorithm repository built with C++ with Python and Matlab bindings.

### JavaScript

* A core technology for the world wide web
* A general-purpose language that extended beyond the browser with the creation of Node.js and other server-side approaches
* NOT related to the Java language
* For DS, the tools based on JavaScript are:
* **TensorFlow.js** makes machine learning and deep learning possible in Node.js and in the browser. TensorFlow.js was also adopted by other open-source libraries including brain.js and machinelearn.js
* **R-js**: The project R-js has re-written linear algebra specifications from the R Language into typescript. This sets the foundation for future projects to implement more powerful math base frameworks like NumPy and SciPy or Python. Typescript is a superset of JavaScript.

### Julia

* Julia was designed at MIT for high-performance numerical analysis and computational science.
* Julia provides speedy development like Python or R while producing programs that run as fast as C or Fortran programs.
* It is compiled which means that Julia 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.
* Julia as a language is only 8 years old, written in 2012, but there is a lot of promise for its future impact on the DS industry.
* One great application of Julia for DS is
* **JuliaDB**: It is a package for working with large persistent data sets.

Module 3

“Libraries for Data Science.”

After watching this video, you will be able to:

* List the scientific computing libraries in Python
* List the visualization libraries in Python
* List the high-level machine learning and deep learning libraries, and
* List the libraries used in other languages

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

* Libraries are a collection of functions and methods that allow you to perform many actions without writing the code.

We will focus on the following Python libraries:

* Scientific Computing Libraries in Python
* Visualization Libraries in Python
* High-Level- Machine Learning and Deep Learning Libraries (High-level means you don’t have to worry about details making studying or improving difficult.)
* Deep Learning Libraries in Python
* Libraries used in other languages

# Scientific Computing Libraries

* Now, scientific computing libraries contain built-in modules providing different functionalities, which you can use directly. They are also called frameworks. For example:
* **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, called a Data Frame. Pandas can also provide easy indexing so you can work with your data.
* **NumPy** libraries are based on *arrays and matrices*, allowing you to apply mathematical functions to the arrays. Pandas is built on top of NumPy.

# Visualization Libraries

You use data visualization methods to communicate with others and display meaningful results of an analysis. These libraries enable you to create graphs, charts, and maps.

* The **Matplotlib** package is the most well-known library for data visualization. They are popular for making graphs and plots, and the graphs are easily customizable.
* **Seaborn**, another high-level visualization library that is based on matplotlib. This library generates heat maps, time series, and violin plots.

# High-Level- Machine Learning and Deep Learning Libraries In Python

* Now, for machine learning, the **Scikit-learn** library contains tools for statistical modeling, including regression, classification, clustering, and so on. It is built on NumPy, SciPy, and matplotlib. It is simple to get started. In this high-level approach, you define the model and specify the parameter types you want to use.
* For building deep learning models, **Keras** allows you to build the standard deep learning model. Like Scikit, the high-level interface allows you to build models in a quick, simple manner. It can function using Graphics processing units (GPU) but in many cases, a lower-level environment is necessary for deep learning.

# Deep Learning Libraries in Python

* **TensorFlow** is a low-level framework used in the large-scale production of deep learning models. It's designed for production and deployment but can be unwieldy for experimentation.
* **Pytorch** is used for experimentation, making it simple for researchers to test ideas.

# Libraries in Other Languages

* **Apache Spark** is a general-purpose cluster-computing framework allowing you to process data using compute clusters. The data is processed in parallel in more than one computer simultaneously. The Spark library has similar functionality to the following:
* Pandas
* Numpy
* Scikit-learn

Apache Spark data processing jobs can be in: Python R Scala, and SQL

* There are many Scala libraries. Scala is predominately used in data engineering and data science. Let’s discuss some ***libraries that are complementary to Spar:***
* **Vegas** is a Scala Library for statistical data visualizations. With Vegas, you can work with data files as well as Spark Data Frames.
* **Big DL** is used for deep learning.
* R has built-in functionality for machine learning and data visualization, but there are also complementary libraries.
* **ggplot2** is a popular library for data visualization in R.
* You can also use libraries that allow you to interface with **Keras** and **TensorFlow**.
* And R was a de-facto standard for open-source data science, but now Python will supersede it.

## Summary:

In this video, you learned that: Libraries usually contain built-in modules providing different functionalities that can be used directly. You can use data visualization methods to communicate with others and display meaningful results of an analysis. For machine learning, the Scikit-learn library contains tools for statistical modeling, including regression, classification, clustering and so on. TensorFlow is a low-level framework used in large-scale production of deep learning models. And, Apache Spark is a general-purpose cluster-computing framework allowing you to process data using compute clusters.

“Application Program Interfaces (API).”

* After watching this video, you will be able to define an API, list API libraries, and define REST API in relation to request and response.

# API Definition

* An application programming interface (API) allows communication between two pieces of software. For example, in a program, you have some data and other software components. You use the API to communicate using inputs and outputs without knowing what happens at the backend.
* The API only refers to the interface. It is the part of the library you see while it contains all the program components.

# List of API Libraries

* To further understand how an API works in a library, consider an example of the Pandas library.
* Pandas is a set of software components where not all components are written in Python. In your program, there is some data and a set of software components. You can use the Pandas API to process the data by communicating with the other software components. The software component at the back end can be the same, but there can be an API for different languages.
* Consider TensorFlow at the backend, written in C++ that can use APIs for other languages, such as: Python JavaScript C++ Java, and Go And thus, the API is just the interface. Other volunteer-developed APIs for TensorFlow are Julia Matlab R Scala And many more.

# REST Libraries

* REST APIs are another popular type of API. The RE stands for Representational. The S stands for State. The T stands for Transfer. They allow you to communicate through the internet and take advantage of resources like storage, data, artificially intelligent algorithms, and much more. In Rest API, your program is the client. The API communicates with a web service you can call through the internet. Though there are rules regarding Communication, Input or Request, and Output or Response.
* Let’s look at some common terms used with regards to API. You or your code is the **client**. The web service is the **resource**. And the client finds the service via an **endpoint**. The client sends **requests** to the resource and receives a **response** from the resource. Data is transmitted over the internet using **HTTP** methods. The Rest APIs get all the information from the request sent by the client. The request is sent using an HTTP message that contains a **JSON file**. The file contains instructions for what operation is to be performed by the web service. This operation is transmitted to the web service via the internet. And the service performs the operation. Similarly, the web service returns a response through an HTTP message, where the information is returned using a JSON file. And this information is transmitted back to the client.
* Now, another example of a Rest API is **Watson Text to Speech API**. This API converts speech to text. In the API call, you will send a copy of the audio file to the API; this is called a **post request**. Then the API will send the text transcription of what the individual is saying. At the backend, the API is making a Get request.
* Finally, let’s look at another example, the **Watson language-translator API**. You send the text you would like to translate into Watson language-translator API. The API will translate the text and send the translation back to you. In this case, the API translates English to Spanish.

## Summary:

In this video, you learned an application programming interface (API) allows communication between two pieces of software, An API is the part of the library you see while the library contains all the components of the program. And REST APIs allow you to communicate through the internet and take advantage of resources like storage, data, artificially intelligent algorithms, and much more.

“Data Sets – Powering Data Science.”

After watching this video, you will be able to

* Define a data set.
* Describe the types of data ownership.
* List the sources of data.
* Describe the Community Data License Agreement.

# Definition of Data Set

* Let’s first define what a dataset is. A data set is a structured collection of data. Data embodies information represented as text, numbers, or media such as images, audio, or video files.
* A **tabular data** set comprises a collection of rows containing 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 a comma separates data values. For example, imagine a dataset of observations from a weather station. Each row represents an observation at a given time, while each column contains information about that 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 format, whereas network data is stored as a graph. For example, the connections between people on a social networking website are often represented as a graph.
* A data set might also include **raw data files**, such as images or audio. The Modified National Institute of Standards and Technology (MNIST) dataset is popular for data science. It contains images of handwritten digits and is commonly used to train image processing systems.

# Types of data ownership

* Traditionally, most data sets were private because they contained proprietary or confidential information such as customer data, pricing data, or other commercially sensitive information. These datasets are typically not shared publicly.
* Over time, many public and private entities such as scientific institutions, governments, organizations, and even companies have started making data sets available to the public as “open data,” providing free information. For example, the United Nations and federal and municipal governments worldwide have published many datasets on their websites, covering the economy, society, healthcare, transportation, the environment, and much more. Access to these and other open datasets enables data scientists, researchers, analysts, and others to uncover previously unknown and potentially valuable insights. They are used to create new applications for commercial purposes and the public good. They are also used to carry out further research. Open data has played a significant role in the growth of data science, machine learning, and artificial intelligence. It has allowed practitioners to hone their skills in various data sets.

# Sources of data

There are many open data sources on the internet. You can find a comprehensive list of available data portals worldwide 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, a popular data science online community, you can find (and contribute) data sets that might be of general interest. Google provides a search engine that might help you find data sets that could be of value to you.

* Open data portal list from around the world

<http://datacatalogs.org/>

* Governmental, intergovernmental and organization websites

<http://data.un.org/> (United Nations)

<http://www.data.gov/> (USA)

<http://www.europeandataportal.eu/en/> (Europe)

* Kaggle

<http://www.kaggle.com/datasets>

* Google data set search

<http://datasetsearch.research.google.com/>

# Community Data License Agreement

Open data distribution and use might be restricted, as defined by certain licensing terms. Without 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 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 choose to share it. In this video, you’ve learned Open data is fundamental to Data Science. Community Data License Agreement makes it easier to share open data, and Open datasets might not meet enterprise requirements, due to the impact they might have on the business.

# Open datasets and sources

In this data-driven world, some datasets are freely available for anyone to access, use, modify, and share. These are called **open datasets**.  
Open datasets include a public license and are very useful for your journey as a Data Scientist. Some of the most informative open dataset sources are listed below.

## **Government Data:**

* <https://www.data.gov/>
* <https://www.census.gov/data.html>
* <https://data.gov.uk/>
* <https://www.opendatanetwork.com/>
* <https://data.un.org/>

## **Financial Data Sources:**

* <https://data.worldbank.org/>
* <https://www.globalfinancialdata.com/>
* <https://comtrade.un.org/>
* <https://www.nber.org/>
* <https://fred.stlouisfed.org/>

## **Crime Data:**

* <https://www.fbi.gov/services/cjis/ucr>
* <https://www.icpsr.umich.edu/icpsrweb/content/NACJD/index.html>
* <https://www.drugabuse.gov/related-topics/trends-statistics>
* <https://www.unodc.org/unodc/en/data-and-analysis/>

## **Health Data:**

* <https://www.who.int/gho/database/en/>
* <https://www.fda.gov/Food/default.htm>
* <https://seer.cancer.gov/faststats/selections.php?series=cancer>
* <https://www.opensciencedatacloud.org/>
* <https://pds.nasa.gov/>
* <https://earthdata.nasa.gov/>
* <https://www.sgim.org/communities/research/dataset-compendium/public-datasets-topic-grid>

## **Academic and Business Data:**

* <https://scholar.google.com/>
* <https://nces.ed.gov/>
* <https://www.glassdoor.com/research/>
* <https://www.yelp.com/dataset>

## **Other General Data:**

* <https://www.kaggle.com/datasets>
* <https://www.reddit.com/r/datasets/>

# Propriety datasets and sources

Proprietary datasets contain data primarily owned and controlled by specific individuals or organizations. This data is limited in distribution because it is sold with a licensing agreement.  
Some data from private sources cannot be easily disclosed, like public data.

National security data, geological, geophysical, and biological data are examples of propriety data. Copyright laws or patents usually bind this type of data. Proprietary datasets that mainly contain sensitive information are less widely available than open datasets.

Some standard propriety dataset sources are listed below.

**Health Care:**

<https://www.sgim.org/communities/research/dataset-compendium/proprietary-datasets>

**Financial Market data:**

<https://datarade.ai/data-categories/proprietary-market-data>

**Google Cloud based datasets:**

<https://cloud.google.com/datasets>

# Dataset licenses

When you select a dataset, it is necessary to look into the license. A license explains whether you can use that dataset or not; or explains if you have to accept certain guidelines to use that dataset. The different license types are listed below.

1. Public domain mark - PUBLIC DOMAIN  
   When a dataset has a Public Domain license, all the rights to use, access, modify and share the dataset are open to everyone. Here there is technically no license.
2. Open data commons public domain dedication and license – PDDL  
   Open Data Commons license has the same features as the Public Domain license, but the difference is the PDDL license uses a licensing mechanism to give the rights to the dataset.
3. Creative commons attribution 4.0 international CC-BY  
   This license allows users to share and modify a dataset, but only if they give credit to the creator(s) of the dataset.
4. Community data license agreement – CDLA PERMISSIVE-2.0  
   Like most open-source licenses, this license allows users to use, modify, adapt, and share the dataset, but only if a disclaimer of warranties and liability is also included.
5. Open data commons attribution license - ODC-BY  
   This license allows users to share and adapt a dataset, but only if they give credit to the creator(s) of the dataset.
6. Creative commons attribution-sharealike 4.0 international - CC-BY-SA  
   This license allows users to use, share, and adapt a dataset, but only if they give credit to the dataset and show any changes or transformations, they made to the dataset. Users might not want to use this license because they have to share the work they did on the dataset.
7. Community data license agreement – CDLA-SHARING-1.0  
   This license uses the principle of ‘copyleft’: users can use, modify, and adapt a dataset, but only if they don’t add license restrictions on the new work(s) they create with the dataset.
8. Open data commons open database license - ODC-ODBL  
   This license allows users to use, share, and adapt a dataset but only if they give credit to the dataset and show any changes or transformations they make to the dataset. Users might not want to use this license because they have to share the work they did on the dataset.
9. Creative commons attribution-noncommercial 4.0 international - CC BY-NC  
   This license is a restrictive license. Users can share and adapt a dataset, provided they give credit to its creator(s) and ensure that the dataset is not used for any commercial purpose.
10. Creative commons attribution-no derivatives 4.0 international - CC BY-ND  
    This license is also a restrictive license. Users can share a dataset if they give credit to its creator(s). This license does not allow additions, transformations, or changes to the dataset.
11. Creative commons attribution-noncommercial-sharealike 4.0 international - CC BY-NC-SA  
    This license allows users to share a dataset only if they give credit to its creator(s). Users can share additions, transformations, or changes to the dataset, but they cannot use the dataset for commercial purposes.
12. Creative commons attribution-noncommercial-noderivatives 4.0 international - CC BY-NC-ND  
    This license allows users to share a dataset only if they give credit to its creator(s). Users are not allowed to modify the dataset and are not allowed to use it for commercial purposes.

***Note: Additional license types exist. Any dataset you use will include details about its license.***

“Sharing Enterprise Data – Data Asset eXchange”

* After watching this video, you will be able to:
* Navigate around IBM's open data repository, the **Data Asset eXchange**.
* Explore open data sets on the Data Asset eXchange.
* Identify the notebook associated with a data set in Watson Studio.
* There are many open data sets available to the public, but it can be difficult to find 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”.
* DAX provides a curated collection of open data sets, both from IBM Research and trusted third-party sources. These data sets are ready for use in enterprise applications, with a wide variety of application types, including images, video, text, and audio.
* DAX aims to foster data sharing and collaboration by keeping data sets available under a Community Data License Agreement (or CDLA). DAX makes it easier for developers to get started with data sets because it provides a single place to access unique, high-quality data sets from trusted sources like IBM Research.
* It also provides **tutorial notebooks** that walk through the basics of data analysis, such as:
* Cleaning the data
* pre-processing of data
* exploratory analysis.
* Certain data sets include **advanced notebooks** that explain how to perform more complex tasks, like:
* creating charts
* training machine-learning models
* Integrating deep learning via the Model Asset eXchange
* Running statistical analysis and time-series analysis.
* The **Data Asset eXchange** and the **Model Asset eXchange** are both available on the IBM Developer website. With these resources, developers can do many important tasks like:
* creating end-to-end analytic and machine learning workflows
* consuming open data and models with confidence under clearly defined license terms
* Now, let’s explore the Data Asset eXchange. Open https://developer.ibm.com/ in your web browser. Then select “Open Source at IBM” From the drop-down, select “Data Asset eXchange”. In the Data Asset eXchange, multiple open data sets are available for you to explore. Let’s say you’ve found a data set that might be very interesting to you: the “NOAA Weather Data - JFK Airport” data set, which contains data from a weather station at the John F. Kennedy Airport in New York. On this data set page, you can click Get this data set to download the NOAA data set from the cloud storage. Run data set notebooks to access the notebooks associated with the data set in Watson. and Preview the data and Notebooks to explore DAX metadata, glossary and the notebook.
* Most data sets on DAX are complemented by one or more Notebooks. Click assets to view all the Jupyter Notebooks and data available. You can then click the source code to view all the notebooks associated with your NOAA project. You can execute all the notebooks in Watson studio to perform data cleaning, pre-processing, and exploratory analysis. If you are already familiar with opening the notebooks in Watson studio, you can log into your IBM Cloud account, create a project, and load all the notebooks into the project. Data sets on DAX also consist of one or more data files. Click the Data option to view the data files, available in your project.

## Summary

In this video, you learned that the IBM Data Asset eXchange (DAX) site contains high-quality open data sets DAX open data sets include tutorial notebooks that provide basic and advanced walk throughs for developers. DAX and MAX are available on the IBM Developer website. You can get, run, and preview data sets and notebooks on DAX, and DAX notebooks are opened in Watson Studio.

“Machine Learning Models – Learning from models to make predictions.”

* After watching this video, you will be able to
* define a machine learning model.
* describe the different learning model types.
* describe how to use a learning model to solve a problem.

# Machine Learning

* Now data contains a wealth of information that can be used to solve certain types of problems. Traditional data analysis approaches can be a person manually inspecting the data or a specialized computer program that automates the human analysis. These approaches reach their limits due to the amount of data to be analyzed or the complexity of the problem. Machine learning (ML) 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,
* Reinforcement Learning.

## Supervised Learning

* The most commonly used type of machine learning is Supervised Learning. In Supervised Learning, a human provides input data and correct outputs. The model tries to identify relationships and dependencies between the input data and the correct output. This type of learning comprises two types of models, **regression** and **classification**.

### Regression Models

* Regression models are used to predict a numeric (or “real”) value. For example, if information is given 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

* Classification models are used to predict whether some information or data belongs to a category (or “class”). For example, for a set of emails along with a designation you can classify whether they are to be considered as spam or not. And so, you can train an algorithm to identify unsolicited emails.

## Unsupervised Learning:

* In Unsupervised Learning, the data is not labeled by a human. The models must analyze the data and try to identify patterns and structure within the data based on its characteristics.
* Clustering is an example of this learning style. Clustering models are used to divide each record of a dataset into one of a similar group.
* 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.
* Another example is anomaly detection that identifies outliers in a dataset, such as fraudulent credit card transactions or suspicious online log-in attempts.

## Reinforcement Learning:

* And the third type of learning, Reinforcement Learning, is loosely based on the way human beings and other organisms learn. So, 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, to get the most rewards 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:

* Deep learning is a specialized type of machine learning. It refers to a general set of models and techniques that 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), **images**, **audio**, **video**, to forecast time series data and much more.
* Deep learning has recently been very successful in these and other areas and hence is becoming an increasingly popular and important tool for data science. It requires large datasets of labeled data to train a model, is compute intensive, and usually requires special purpose hardware to achieve acceptable training times.
* Now 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**. The learning frameworks 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. Let’s briefly outline the high-level tasks involved in building a model 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. For example, you can draw bounding boxes around objects and label 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 can 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.

## Summary

In this video you learned that: Machine learning (ML) uses algorithms – also known as “models” ‒ to identify patterns in the data. The process by which the model learns data patterns is called “model training”. Types of ML are Supervised, Unsupervised, and Reinforcement. Supervised learning comprises two types of models, regression and classification. And deep learning refers to a general set of models and techniques that loosely emulate the way the human brain solves a wide range of problems.

The Model Asset Exchange

After watching this video, you will be able to

* navigate the Model Asset Exchange from IBM Research
* explain how deep learning model-serving detects images.
* The Model Asset eXchange, or “MAX”, on the IBM Developer platform, is a free open-source resource for deep learning models.

# MAX Reduces Time to Value

* The tasks needed to train a model from scratch require a large amount of data, labor, time, and resources. Because of this, 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.
* Models are created by running data through a Model using compute resources and domain expertise. After research, evaluation, test, train and validate steps are complete, you will have a validated model.
* 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 deep learning models to solve common business problems. These models have been fully 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, which reduces the risk of legal liabilities.
* On MAX, you can find models for a variety of domains, including:
* Object detection,
* Image, audio, video, and text classification,
* Named entity recognition,
* Image to text translation,
* Human pose detection, and more.

# Typical model-serving microservice

* Let’s look at the components of a typical model-serving microservice. Each microservice includes:
* a pre-trained deep learning model
* code that pre-processes **the input** before it is analyzed by the model
* code that post-processes **the model output**
* a standardized public API that makes the services functionality available to applications.
* Model-serving microservices are created by running inputs through a validated model and then applying the output to a rest API.
* After implement, package, document, and test steps are complete, you will have a model-serving microservice that can then be sent to a Local machine, or a Private, Hybrid, or Public cloud.
* MAX model-serving microservices are built and distributed as open-source **Docker** images. Docker is a container platform that makes it easy to build and deploy applications.
* The Docker image source is published on GitHub and can be downloaded and customized for use in personal and commercial environments.
* Use the **Kubernetes** open-source system to automate the deployment, scaling, and management of these Docker images.
* **Red Hat OpenShift** is a popular enterprise-grade Kubernetes platform. It is available on:
* IBM Cloud
* Google Cloud Platform
* Amazon Web Services
* Microsoft Azure
* Let’s explore some machine learning models. Go to ml-exchange.org. Here you can view and use multiple predefined models.
* We'll explore the predefined object detector model. This model will recognize objects in an image because it consists of:
* a deep convolutional net base model for image feature extraction
* and added convolutional layers specialized in object detection.
* On the MAX object detector page, select CodePen. CodePen is an online tool used by developers to edit front-end languages like HTML, JavaScript, and CSS.
* You will be redirected to the CodePen page, where you can select MAX Tensorflow.js model. This model is trained to identify objects in an image and assigns each pixel of the image to a particular object. Here you can upload different images of a person, dog, cat, truck, or car.
* The model was previously trained on labeled images, so now it can recognize images even when they are not labeled.
* Select an image to see what happens when the model invokes the prediction endpoint.
* Click on Extract prediction. This invokes the prediction endpoint, and the image is uploaded.
* The prebuilt TFJS model prepares the input image for pre-processing.
* The deep learning model algorithm identifies the different objects in the image. It generates its response using the prediction results and returns the result to the application.
* You will see the existing image separated into two different images:
* the background image and
* the image of the dog.
* The model test is complete. You have confirmed that this model is able to identify items within an image without using predefined labels.

## Summary

In this video, you learned: The Model Asset eXchange is a free open source repository for ready-to-use and customizable deep learning microservices. To reduce time to value, consider taking advantage of pre-trained models for certain types of problems. MAX model-serving microservices are built and distributed on GitHub as open source Docker images. Red Hat OpenShift is a Kubernetes platform used to automate deployment, scaling, and management of microservices. Ml-exchange.org has multiple predefined models. The CodePen tool lets users edit front-end languages.

Module 4

“Introduction to Jupyter Notebooks.”

* After watching this video, you will be able to
* define a Jupyter Notebook
* explain how to use JupyterLab
* describe how to use the notebooks in JupyterLab.

# Jupyter Notebook

* Jupyter Notebooks originated as “**iPython**,” originally developed for Python programming. Later, when it started supporting additional languages, it was renamed Jupyter, which stands for **Julia**, **Python**, and **R**. However, now, it supports many other languages.
* A Jupyter Notebook is a **browser-based application** that allows you to create and share documents containing **code**, **equations**, **visualizations**, **narrative text links**, and more.
* It is like a scientist’s lab notebook, where a scientist records all steps to perform their experiments and the results they can reproduce. In the same way, a Jupyter Notebook allows a Data Scientist to record their data experiments and results that others can reuse.
* Now a Jupyter Notebook file allows you to combine descriptive text, code blocks, and code output in a single file. When you run the code, it generates the output, including plots and tables, within the notebook file. And you can then export the notebook to a **PDF** or **HTML file format** that can then be shared with anyone.

# Jupyter Lab

* Next, let’s learn about Jupyter Lab. Jupyter Lab is a **browser-based application** that allows you to access multiple Jupyter Notebook files, other code, and data files.
* In addition, it extends the functionalities of Jupyter Notebooks by enabling you to work with multiple notebooks, text editors, terminals, and custom components in a flexible, integrated, and extensible manner.
* It is **compatible** with several file formats like **CSV**, **JSON**, **PDF**, **Vega**, and so on. And it is also an open source.
* Jupyter Notebooks can be used with **cloud-based services like IBM and Google Colab**. They don't require any installation on your local machine.
* They give you access to the Jupyter Notebook environment and allow you to import and export notebooks using the standard **IPython** Notebook file format. Also, these services support the Python language and other languages as well.
* Jupyter Notebooks can be installed via the command line using the pip install function. It can also be downloaded locally on your laptop through the **Anaconda Platform** from Anaconda dot com. Anaconda is one of the popular distributions which includes Jupyter and Jupyterlab.
* So, for this course, you have access to a hosted version of JupyterLab in Skills Network Labs, so you do not require any installations on your own device to complete the hands-on labs. As shown here, you will see a screen that will launch the Jupyter Lab in the virtual environment. Simply click the Open tool tab.

## Summary

In this video, you learned that Jupyter Notebooks are used in Data Science for recording experiments and projects. Jupyter Lab is compatible with many files and Data Science languages. And there are different ways to install and use Jupyter Notebooks.

“Getting started with Jupyter”

* After watching this video, you will be able to:
* Describe how to run, insert, and delete a cell in a notebook.
* Work with multiple notebooks.
* Present the notebook, and shut down the notebook session.

# Hands On to Jupyter Notebook

* In the lab session of this module, you can launch a notebook using the Skills Network virtual environment. After selecting the check box, click the Open tool tab, and the environment will open the Jupyter Lab. Here you see the open notebook. On opening the notebook, you can change the name of the notebook. Click File. Then click Rename Notebook to give the required name. And you can now start working on your new notebook. In the new notebook, print “hello world”.
* Then click the Run button to show that the environment is giving the correct output. On the main menu bar at the top, click Run. In the drop-down menu, click Run Selected Cells to run the current highlighted cells. Alternatively, you can use a shortcut, press Shift + Enter. In case you have multiple code cells, click Run All cells to run the code in all the cells. You can add code by inserting a new cell. To add a new cell, click the plus symbol in the toolbar. In addition, you can delete a cell. Highlight the cell and on the main menu bar, click Edit, and then click Delete Cells. Alternatively, you can use a shortcut by pressing D twice on the highlighted cell. Also, you can move the cells up or down as required. So, now you have learned to work with a single notebook. Next, let’s learn to work with multiple notebooks. Click the plus button on the toolbar and select the file you want to open. Another notebook will open. Alternatively, you can click File on the menu bar and click Open a new launcher or Open a new notebook. And when you open the new file, you can move them around. For example, as shown, you can place the notebooks side by side. On one notebook, you can assign variable one to the number 1, and variable two to the number 2 and then you can print the result of adding the numbers one and two.
* As a data scientist, it is important to communicate your results. Jupyter supports presenting results directly from the notebooks. You can create a Markdown to add titles and text descriptions to help with the flow of the presentation. To add markdown, click Code and select Markdown. You can create line plots and convert each cell and output into a slide or sub-slide in the form of a presentation.
* The slides functionality in Jupyter allows you to deliver code, visualization, text, and outputs of the executed code as part of a project.
* Now, when you have completed working with your notebook or notebooks, you can shut them down. Shutting down notebooks release their memory. Click the stop icon on the sidebar, it is the second icon from the top. You can terminate all sessions at once or shut them down individually. And after you shut down the notebook session, you will see “no kernel” at the top right. This confirms it is no longer active, you can now close the tabs.

# Some Shortcuts for Working in Jupyter Notebook

* If the cell is “CODE” and you want to convert it into “MARKDOWN”, then the keyboard shortcut is [Press ESC and then M]. The code will automatically be converted into Markdown.
* Within markdown, for heading of first level, add one # sign and then space before the phrase. For second level heading, add two ## sign and then space and so on uptil six ###### signs and space before the phrase.
* For bold, type two asterisk (\*\*) or two underscore signs (\_\_) before and after the phrase without space.
* For italics, type one asterisk (\*) or one underscore sign (\_) before and after the phrase without space.
* For both bold and space, add three asterisks (\*\*\*) or underscore signs (\_\_\_) before and after the phrase without space.
* For table, type | Heading | 2nd heading | 3rd heading |

| ------------ | --------------- | --------------- |

| values | values | values |

* For ordered list: add 1. Or 2. Or 3. Before the items while typing each item in a new line.
* For un-ordered list: add \* or + before the items while typing each item in a new line.
* For hyperlinking a web address: [Name of Website](past URL)

For images: image:! [Image Name](past URL).

## Summary

In this video, you learned how to: Run, delete, and insert a code cell. Run multiple notebooks at the same time. Present a notebook using a combination of Markdown and code cells. And shut down your notebook sessions after you have completed your work.

“Jupyter Kernels.”

# Introduction to Kernels

* After watching this video, you will be able to define a kernel, and describe how to work with kernels.
* A notebook kernel is a **computational engine** that executes the code contained in a Notebook file. Jupyter Kernels for many languages exist, and we will explore some that are relevant in Data Science.
* When a Notebook document opens, the related kernel launches automatically. When the Notebook is executed, the kernel performs the computation and produces the results. Depending on your settings, you may need to install other notebook languages in your Jupyter environment.
* In the Skills Network lab environment, a few languages have been pre-installed for you. The first one is the Python kernel.
* When you launch a notebook, pick the language you are interested in for your Data Science project. The Python kernel allows you to run python cells.
* You can run the Python script in the cells to produce an output. The top right corner of the Notebook shows the name of the kernel. Here it shows the Python kernel. You have the option to run other kernels.
* The Skills Network virtual Jupyter environment has Apache, Julia, R, and Swift. You can use any language to execute your code, either by selecting the kernel on the launch page or clicking the top right icon and selecting the kernel from the drop-down menu.
* If running the kernel on your local machine, you will need to manually install the languages through your command line interface (CLI).

## Summary

* In this video, you learned that The kernel acts like a computational engine and executes the code in a Notebook file. Jupyter Notebook supports different languages, and you can switch to a different kernel as per your requirement.

“Jupyter Architecture”

* After watching this video, you will be able to describe the basic Jupyter architecture, and explain Jupyter architecture for conversion of a file format.

# Client and Kernel

* Jupyter architecture implements a **two-process model** with a **kernel** and a **client**.
* The client is the interface offering the user the ability to send code to the kernel. It is the browser in a Jupyter Notebook.
* The kernel executes the code and returns the result to the client for display.
* Jupyter Notebooks represent your code, metadata, contents, and outputs.

# Saving a Jupyter Notebook

* When you save the Notebook, it is sent from your browser to the Notebook server.
* It saves the notebook file on a disk as a **JSON file** with a **.ipynb** (pronounced as dot i PI NB) extension.
* The Notebook server is responsible for saving and loading the notebooks. And the kernel executes the cells of code contained in the Notebook when the user runs them.
* The Jupyter architecture uses the NB convert tool to convert files to other formats. For example, if we want to convert a notebook file into an HTML file, the notebook is first modified by a **preprocessor**, then an **exporter** converts the notebook to the new file format. Finally, a **postprocessor** will work on the exported file to give the final output. After conversion, on giving the url of the file, the HTML file displays.

## Summary

In this video, you learned that: Jupyter implements a two-process model with a kernel and a client. The Notebook server is responsible for saving and loading the notebooks. The kernel executes the cells of code contained in the Notebook. And the Jupyter architecture uses the NB convert tool to convert files to other formats.

Additional Anaconda Jupyter Environments

* After watching this video, you will be able to:
* describe Anaconda and its data science features
* describe Anaconda Jupyter environments
* identify tools in Anaconda Jupyter environments.

# Computational Notebooks

* Computational notebooks combine code, computational output, explanatory text, and multimedia resources into a single document.
* Jupyter Notebook is a popular type of computational notebook because it supports dozens of programming languages.
* JupyterLab and VS Code are popular environments for creating and modifying Jupyter Notebooks on a local device.

## Jupyter Lab

* JupyterLab is an open-source, web-based application based on Jupyter Notebook. You can create code, interactive visualizations, text, and equations, just like with Jupyter Notebook.
* JupyterLab includes expanded features with some of Anaconda's most extensive pre-installed Python libraries, including NumPy, Pandas, and Matplotlib.

## Anaconda

* Anaconda is a free and open-source distributor for Python and R, the top languages used in data science and machine learning.
* Anaconda has fifteen hundred plus libraries. It is free to install and has free community support for any users who need help with Python.
* The downloadable Anaconda Navigator graphical user interface allows users to install new packages on their local device without using a command line interface or ‘CLI.’ You can download Anaconda Navigator from the given URL ([www.anaconda.com](http://www.anaconda.com)).
* Here is the home page of the Anaconda Navigator. To launch JupyterLab, click Launch in the JupyterLab box. If the Launch button is missing, click Install first, and then click Launch.
* To start with the Jupyter Notebook, type Jupyter Notebook(anaconda3) in the search bar and press enter. The JupyterLab dashboard opens in the browser on the localhost. It is specifically designed to manage Jupyter Notebooks.
* To create a new Jupyter Notebook, click New and select Python 3. This opens a new notebook in a new tab. You will see the URL, which shows the filename and the kernel. It also shows the Last Checkpoint. Let’s rename your notebook by clicking Untitled. Type a name for the notebook and click Rename.
* Next, you will review two main cell types: Code and Markdown. In the dropdown menu, select Code.
* A code cell contains code to be executed in the kernel and displays its output. To execute the cell, click Run.
* Alternatively, in the dropdown menu, select Markdown. A Markdown cell contains rich text and displays its output in place when it executes.
* To download a notebook, go to File and click Download as. You will see several download options. You can select the option you want.

## VisualStudio Code

* VS Code is a free, open-source code editor for debugging and task-running operations.
* VS code works on Linux, Windows, and macOS.
* It supports multiple languages, syntax highlighting, auto-indentation, and more. VS Code is one of the most popular development environment tools.
* If you prefer to install VS code separately, without using Anaconda Navigator, you can go to code.visualstudio.com, click the download option that applies to your device, then follow the install instructions. A separate installation of VS Code will work the same as in Anaconda Navigator, but it will not configure for Anaconda, Python, or Jupyter Notebooks. To open VS Code using Anaconda Navigator, open Anaconda Navigator, find the VS Code application, and click Launch. Once installed, you will see the Get Started screen. You need to install a few extensions to execute Python code in VS Code. First, click Extensions or use Ctrl + Shift + X keys to open Extensions. Then search for “Python”; all the extensions related to Python will appear. Once you install the extensions, click File. Then select New File. In New File, select Jupyter Notebook. The notebook will look like this. Notice that the kernel is Python. Write your code and then execute it using the RUN icon. You will get a confirmation that your code has been executed successfully. And finally, navigate to File and select Save.

## Summary

In this video, you learned that Jupyter is a popular computational notebook tool because it supports dozens of programming languages. The Anaconda Navigator GUI can launch multiple applications on a local device. Jupyter environments in the Anaconda Navigator include JupyterLab and VS Code. And you can download Jupyter environments separately from the Anaconda Navigator, but they may not be configured properly.

“Additional Cloud-Based Jupyter Environments”

* After watching this video, you will be able to:​
* describe cloud-based Jupyter environments and their data science features​
* navigate cloud-based Jupyter environments, and​ identify tools in cloud-based environments.​

# Computational notebooks

* Computational notebooks combine code, computational output, explanatory text, and multimedia resources in a single document. ​
* Jupyter notebook is a popular type of computational notebook because it supports dozens of programming languages. ​
* Popular cloud-based environments used to create and modify Jupyter notebooks include:​
* JupyterLite
* GoogleColaboratory​.​

# JupyterLite

* JupyterLite is a lightweight tool built from JupyterLab components ​that executes entirely in the browser.​
* JupyterLite does not require a dedicated Jupyter server. ​ Only a web server is required, which means ​we can deploy JupyerLite as a static website.​
* We can also use it to create interactive graphics and visualizations because it supports many visualization libraries like **Altair**, **Plotly**, and **ipywidgets**​.
* Since JupyterLite is a distribution of JupyterLab, ​it includes the latest improvements and features.​ To launch JupyterLite, open a browser and type jupyter.org/try-jupyter/lab in the URL field. Then press Enter.​ JupyterLite will appear.​ Next, click Python(Pyodide). ​​ Here is a view of a JupyterLite notebook. We know this is a JupyterLite notebook because ​we see the kernel is Python Pyodide. ​ This kernel allows installing and running Python packages in a browser. You will notice different kernels depending on the type of Jupyter environment you use.
* For cloud based Jupyter environments, **Python Pyodide** and **Python Pyolite** are common kernels. ​
* The default kernel for JupyterLite is​ Pyolite. ​ Pyolite is a Python kernel based on Pyodide. Pyolite runs in the background, so that intensive computations can execute quickly. ​ Other kernels can also be used with JupyterLite.​

# Google Colaboratory

* Google Colaboratory (or 'GoogleColab') is a free Jupyter notebook environment that runs entirely in the cloud.​ GoogleColab Jupyter notebooks execute on a **browser**, and GoogleColab projects are stored on **Google Drive** and **GitHub**. ​
* You can upload and share notebooks without setup and installation.​ You can also clone projects from GitHub and execute them in GoogleColab.​ Most machine learning and visualization libraries are pre-installed, like **scikit-learn** and **matplotlib**.​
* With GoogleCollab, you can develop many trending data science applications **“on the fly”**, which is to say, quickly without a lot of setup or preparation.​
* To open the Colab notebook, open Google Drive, and​ click New​​ To explore GoogleColab,​ from the Google Drive menu, select More. ​Then select Google Colaboratory.​​ The GoogleColab notebook will appear.​ In the notebook, write the code ​in the code section, and then to execute the code,​ click the Run icon.​ To add more Code or Text cells, you need to click ​+Code and​ +Text. ​ Here, text cells are used to write rich text, or you can set these cells as Markdown cells.​

## Summary

In this video, you learned that:​ Jupyter is a popular computational notebook tool because it supports dozens of programming languages.​ The Anaconda Navigator GUI can launch multiple applications.​ Additional open-source Jupyter environments include the following: JupyterLab, JupyterLite, VS Code, and Google Colaboratory. ​ JupyterLite is a browser-based tool.​

Module 5

“Introduction to R and RStudio”

* After watching this video, you will be able to explain
* what is R
* list R capabilities
* describe RStudio environment
* list the R libraries for data science.

# What is R?

* R is a **statistical programming language**. It is a powerful tool for:
* data processing
* manipulation,
* statistical inference,
* data analysis,
* machine learning algorithm.
* Based on 2017 analysis, it was found that R is used most by **academics**, **healthcare**, and the **government**.
* R supports importing of data from different sources like flat files, databases, web, and statistical software such as SPSS and STATA.
* R is a preferred language for some data scientists because R functions are easy to use.
* It is also known for producing great visualizations and contains packages to handle data analysis without the need to install additional libraries.

# RStudio

* A popular integrated development environment for developing and running the R language source code and programs is RStudio. It improves and increases productivity with the R language.
* R studio includes:
* a syntax-highlighting editor that supports direct code execution and a place where you can keep a record of your work,
* a Console for typing R commands,
* a workspace and History tab that shows the list of R objects you created during your R session and the history of all previous commands,
* Files, Plots, Packages, and Help tabs.

## Tabs in RStudio

* The Files tab shows files in your working directory.
* The Plots tab displays the history of plots you have created. You can also export plots to PDF or image files.
* The Packages tab displays external R packages available on your local computer.
* And, the Help tab provides help on R resources, R studio support, packages, and many more.

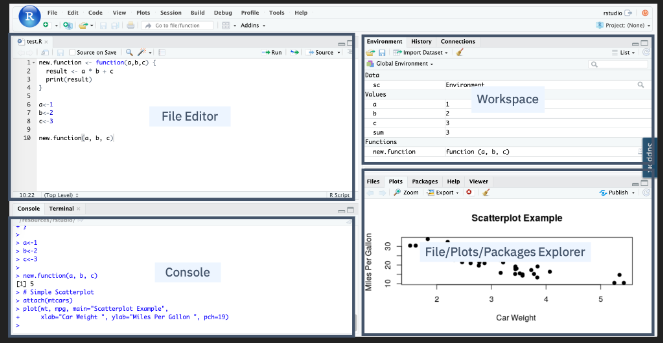
## Popular R Libraries for Data Science

* If R is your tool choice for data science, here are some popular R libraries available in the Data Science community:
* **dplyr** for manipulating data,
* **stringr** for manipulating strings,
* **ggplot** for visualizing data, and
* **caret** for machine learning.
* To get you up and learning quickly, we have provided you with an R Studio virtual environment as part of the Skills Network Labs. This virtual lab environment is designed to assist you to easily practice what you learn in the course and skip the need to create an account or download or install anything.

## Summary

In this video, you learned the capabilities of R and its uses in Data Science, the RStudio interface for running R codes, and popular R packages for Data Science.

# Some Introductory Key Points for Working with RStudio

* In the **Console panel**, you can quickly try some R commands and see the results immediately.
* In the **File Editor** panel, you can write your R code or other text files with the help of syntax highlighting and auto completion.
* In the **Workspace panel**, you can review and manage the created objects.
* In the **File/Plots/Package** Explorer panel, you can manage your files and other assets, such as plots or packages.
* To clear console, press Ctrl + L
* To assign a value of 10 to a variable “x”, we will write the code as follow:
* X <- 10

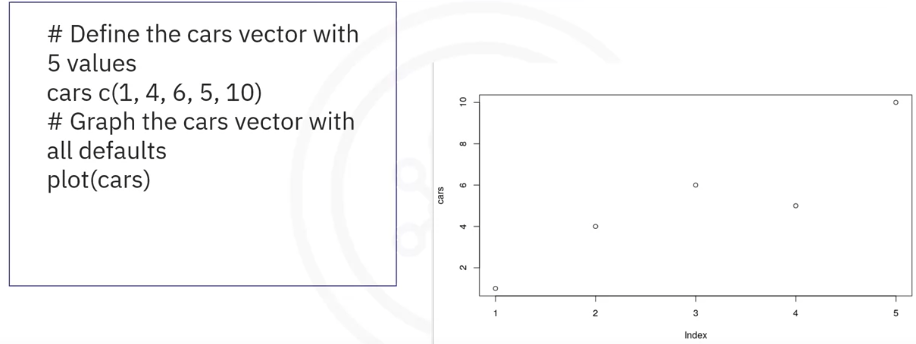
“Plotting in RStudio”

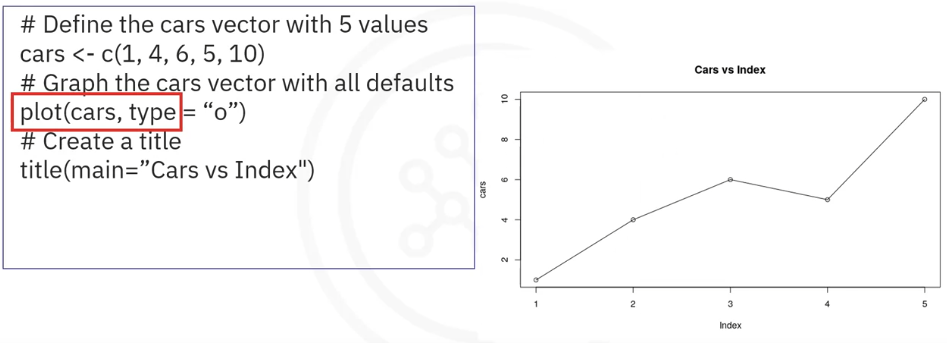
* After watching this video, you will be able to:
* List the R data visualization packages,
* Use the inbuilt R plot function,
* Use the R ggplot library to add functions and arguments to the plot
* Add titles and names to the plot.

# Using Data Visualization in R

* With the influx of data, one of your many jobs as data scientists is to produce insights using visualizations.
* R has different packages for data visualization that you can use based on your requirement.
* To install these packages in your R environment, use the install.packages and the package name command. Examples of R packages include the following.
* ggplot is used for data visualizations such as histograms, bar charts, scatterplots, and so on. It allows adding layers and components to a single visualization.
* Plotly is used for web-based data visualizations that can be displayed or saved as individual HTML files.
* Lattice is used to implement complex, multi-variable data sets. It is a high-level data visualization library that can handle graphics without customizations
* Leaflet is used for creating interactive plots.

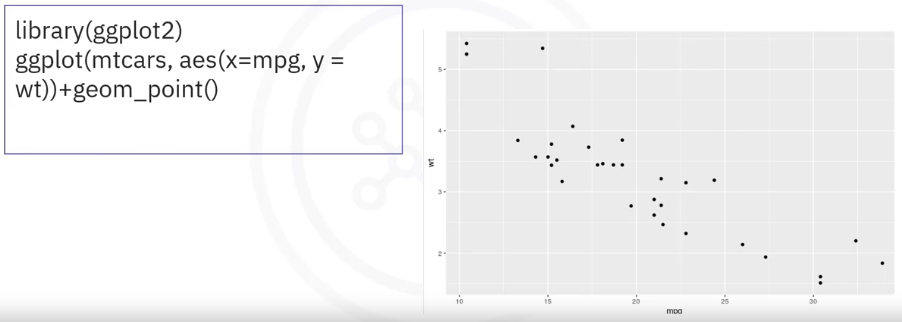
# Using the Plot Function

* R has inbuilt functions to create plots and visualization. For example, you can create a plot using the definition shown here. The plot function returns a scatterplot of the values vs. the index.
* You can also add lines to the function and a title to make the visualization easier to read and understand. To add a line, you specify the type and to add a title, you select the title function. In the plot, you have added a line and a title.

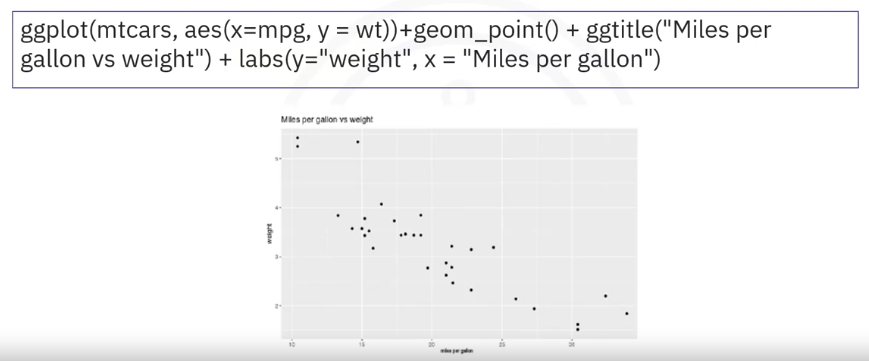


# Using ggplot

* You can create informative visualizations using the ggplot library of R. It can handle complex requests by adding layers to plots using different functions and arguments. For example, to create a scatter plot, let’s use the inbuilt dataset Mtcars. You will first read the ggplot library into the memory using the library function. Next, use the ggplot function on the data frame MTcars, specify the X axis as miles per gallon and the Y axis as weight. Then add the geom point function to specify a scatter plot; otherwise, it will return an empty plot. The output will be an easier-to-read plot.



* In addition, you can add titles and change the axis name by using the Ggtitle argument and the lab’s argument to specify appropriate names for both axes. The result will be a graph with meaningful titles.



* In the lab, you will recreate the graphics with ggplot and the extension library called GGally. GGally extends ggplot by adding several functions to reduce the complexity of combining geometric objects with transformed data. In this video, you learned about: Popular data visualization packages in R, Plotting with the inbuilt R plot function, Plotting with ggplot, Adding titles and changing the axis names using the ggtitle and lab’s function.

“Overview of Git/GitHub”

* In this video, you’ll get an overview of Git and GitHub, which are popular environments among developers and data scientists for performing version control of source code files and projects and collaborating with others.

# Version Control

* You can’t talk about Git and GitHub without a basic understanding of what version control is.
* A version control system allows you to keep track of changes to your documents. This makes it easy for you to recover older versions of your document if you make a mistake, and it makes collaboration with others much easier. Here is an example to illustrate how version control works. Let’s say you’ve got a shopping list and you want your roommates to confirm the things you need and add additional items. Without version control, you’ve got a big mess to clean up before you can go shopping. With version control, you know exactly what you need after everyone has contributed their ideas.

Git is free and open source software distributed under the GNU General Public License. Git is a distributed version control system, which means that users anywhere in the world can have a copy of your project on their own computer. When they’ve made changes, they can sync their version to a remote server to share it with you. Git isn’t the only version control system out there, but the distributed aspect is one of the main reasons it’s become one of the most common version control systems available. Version control systems are widely used for things involving code, but you can also version control images, documents, and any number of file types. You can use Git without a web interface by using your command line interface, but GitHub is one of the most popular web-hosted services for Git repositories. Others include GitLab, BitBucket, and Beanstalk. There are a few basic terms that you will need to know before you can get started. The SSH protocol is a method for secure remote login from one computer to another. A repository contains your project folders that are set up for version control. A fork is a copy of a repository. A pull request is the way you request that someone reviews and approves your changes before they become final. A working directory contains the files and subdirectories on your computer that are associated with a Git repository. There are a few basic Git commands that you will always use. When starting out with a new repository, you only need create it once: either locally, and then push to GitHub, or by cloning an existing repository by using the command "git init".

"git add" moves changes from the working directory to the staging area. "git status" allows you to see the state of your working directory and the staged snapshot of your changes. "git commit" takes your staged snapshot of changes and commits them to the project. "git reset" undoes changes that you’ve made to the files in your working directory. "git log" enables you to browse previous changes to a project. "git branch" lets you create an isolated environment within your repository to make changes. "git checkout" lets you see and change existing branches. "git merge" lets you put everything back together again. To learn how to use Git effectively and begin collaborating with data scientists around the world, you will need to learn the essential commands. Luckily for us, GitHub has amazing resources available to help you get started. Go to try.github.io to download the cheat sheets and run through the tutorials. In the following modules, we'll give you a crash course on setting up your local environment and getting started on a project.