Week 1 Module 3

Video 3.1. In this video, we will review several data science libraries.

Libraries are a collection of functions and methods that enable you to perform a wide variety of actions without writing the code yourself.
We will focus on Python libraries: Scientific Computing Libraries in Python Visualization Libraries in Python High-Level Machine Learning and Deep Learning Libraries – "High-level" simply means you don't have to worry about details, although this makes it difficult to study or improve Deep Learning Libraries in Python Libraries used in other languages
Libraries usually contain built-in modules providing different functionalities that you can use directly; these are sometimes called "frameworks."
There are also extensive libraries, offering a broad range of facilities.
Pandas offers data structures and tools for effective data cleaning, manipulation, and analysis.
It provides tools to work with different types of data.
The primary instrument of Pandas is a two-dimensional table consisting of columns and rows.
This table is called a "DataFrame" and is designed to provide easy indexing so you can work with your data.
NumPy libraries are based on arrays, enabling you to apply mathematical functions to these arrays.
Pandas is actually built on top of NumPy
Data visualization methods are a great way to communicate with others and show the meaningful results of analysis.
These libraries enable you to create graphs, charts and maps.
The Matplotlib package is the most well-known library for data visualization, and it's excellent for making graphs and plots.
The graphs are also highly customizable.
Another high-level visualization library, Seaborn, is based on matplotlib.
Seaborn makes it easy to generate plots like heat maps, time series, and violin plots.
For machine learning, the Scikit-learn library contains tools for statistical modeling, including regression, classification, clustering and others.
It is built on NumPy, SciPy, and matplotlib, and it's relatively simple to get started.
For this high-level approach, you define the model and specify the parameter types you would like to use.
For deep learning, Keras enables you to build the standard deep learning model.
Like Scikit-learn, the high-level interface enables you to build models quickly and simply.
It can function using graphics processing units (GPU), but for many deep learning cases a lower-level environment is required.
TensorFlow is a low-level framework used in large scale production of deep learning models.
It's designed for production but can be unwieldy for experimentation.
Pytorch is used for experimentation, making it simple for researchers to test their ideas
Apache Spark is a general-purpose cluster-computing framework that enables you to process data using compute clusters.
This means that you process data in parallel, using multiple computers simultaneously.
The Spark library has similar functionality as
Pandas Numpy
Scikit-learn
Apache Spark data processing jobs can use Python
R Scala, or SQL

There are many libraries for Scala, which is predominately used in data engineering
but is also sometimes used in data science.
Let’s discuss some of the libraries that are complementary to Spark
Vegas is a Scala library for statistical data visualizations.
With Vegas, you can work with data files as well as Spark DataFrames.
For deep learning, you can use BigDL.
R has built-in functionality for machine learning and data visualization, but there are also
several complementary libraries: ggplot2 is a popular library for data visualization
in R. You can also use libraries that enable you
to interface with Keras and TensorFlow.
R has been the de-facto standard for open source data science but it is now being superseded
by Python.

Video 3.2. In this video we will discuss Application Programming Interfaces, or APIs. Specifically,

we will discuss: What an API is, API Libraries, REST APIs, including:
Request and Response. An API lets two pieces of software talk to each other. For example
you have your program, you have some data, you have other software components. You use
the API to communicate with the other software components.You don’t have to know how the
API works, you just need to know its inputs and outputs. Remember, the API only refers
to the interface, or the part of the library that you see. The “library” refers to
the whole thing. Consider the pandas library. Pandas is actually a set of software components,
many of which are not even written in Python. You have some data.
You have a set of software components. We use the pandas API to process the data
by communicating with the other software components. There can be a single software component at
the back end, but there can be a separate API for different languages. Consider TensorFlow,
written in C++. There are separate APIs in Python, JavaScript,
C++ Java, and Go. The API is simply the interface.
There are also multiple volunteer-developed APIs for TensorFlow; for example Julia, MATLAB,
R, Scala, and many more. REST APIs are another popular type of API. They enable you to communicate
using the internet, taking advantage of storage, greater data
access, artificial intelligence algorithms, and many other resources. The RE stands for
“Representational,” the S stands for “State,” the T stand for “Transfer.” In rest APIs,
your program is called the “client.” The API communicates with a web service that you
call through the internet. A set of rules governs Communication, Input or Request, and
Output or Response. Here are some common API-related terms. You or your code can be thought of
as a client. The web service is referred to as a resource. The client finds the service
through an endpoint. The client sends the request to the resource and the response to
the client. HTTP methods are a way of transmitting data over the internet We tell the REST APIs
what to do by sending a request. The request is usually communicated through an HTTP message.
The HTTP message usually contains a JSON file, which contains instructions for the operation
that we would like the service to perform. This operation is transmitted to the web service
over the internet. The service performs the operation. Similarly, the web service returns
a response through an HTTP message, where the information is usually returned using
a JSON file. This information is transmitted back to the
client. The Watson Speech to Text API is an example

of a REST API. This API converts speech to text. In the API call, you send a copy of
the audio file to the API; this process is called a post request. The API then sends
the text transcription of what the individual is saying. The API is making a get request.
The Watson Language-Translator API provides another example. You send the text you would
like to translate into the API, the API translates the text and sends the translation back to
you. In this case we translate English to Spanish. In this video, we've discussed
what an API is, API Libraries, REST APIs, including Request and Response. Thank you
for watching this video.

Video3.3 In this video we'll discuss data sets: what they are, why they are important in data science,

and where to find them.
Let's first loosely define what a data set is.
A data set is a structured collection of data.
Data embodies information that might be represented as text, numbers, or media such as images, audio, or video files.
A data set that is structured as tabular data comprises a collection of rows, which in turn
comprise columns that store the information.
One popular tabular data format is "comma separated values," or CSV.
A CSV file is a delimited text file where each line represents a row and data values
are separated by a comma.
For example, imagine a data set of observations from a weather station.
Each row represents an observation at a given time, while each column contains information
about that particular observation, such as the temperature, humidity, and other weather
conditions.
Hierarchical or network data structures are typically used to represent relationships
between data.
Hierarchical data is organized in a tree-like structure, whereas network data might be stored
as a graph.
For example, the connections between people on a social networking website are often represented
in the form of a graph.
A data set might also include raw data files, such as images or audio.
The MNIST dataset is popular for data science.
It contains images of handwritten digits and is commonly used to train image processing
systems.
Traditionally, most data sets were considered to be private because they contain proprietary
or confidential information such as customer data, pricing data, or other commercially
sensitive information.
These data sets are typically not shared publicly.
Over time, more and more public and private entities such as scientific institutions,
governments, organizations and even companies have started to make data sets available to
the public as "open data," providing a wealth of information for free.
For example, the United Nations and federal and municipal governments around the world
have published many data sets on their websites, covering the economy, society, healthcare,
transportation, environment, and much more.
Access to these and other open data sets enable data scientists, researchers, analysts, and
others to uncover previously unknown and potentially useful insights.
They can create new applications for both commercial purposes and the public good.
They can also carry out new research.
Open data has played a significant role in the growth of data science, machine learning,
and artificial intelligence and has provided a way for practitioners to hone their skills

on a wide variety of data sets.
There are many open data sources on the internet.
You can find a comprehensive list of open data portals from around the world on the
Open Knowledge Foundation’s datacatalogs.org website.
The United Nations, the European Union, and many other governmental and intergovernmental
organizations maintain data repositories providing access to a wide range of information.
On Kaggle, which is a popular data science online community, you can find and contribute
data sets that might be of general interest.
Last but not least, Google provides a search engine for data sets that might help you find
the ones that have particular value for you.
It’s important to recognize that open data distribution and use might be restricted,
as defined by its licensing terms.
In absence of a license for open data distribution, many data sets were shared in the past under
open source software licenses.
These licenses were not designed to cover the specific considerations related to the
distribution and use of data sets.
To address the issue, the Linux Foundation created the Community Data License Agreement,
or CDLA.
Two licenses were initially created for sharing data: CDLA-Sharing and CDLA-Permissive.
The CDLA-Sharing license grants you permission to use and modify the data.
The license stipulates that if you publish your modified version of the data you must
do so under the same license terms as the original data.
The CDLA-Permissive license also grants you permission to use and modify the data.
However, you are not required to share changes to the data.
Note that neither license imposes any restrictions on results you might derive by using the data,
which is important in data science.
Let’s say, for example, that you are building a model that performs a prediction.
If you are training the model using CDLA-licensed data sets, you are under no obligation to
share the model, or to share it under a specific license if you do choose to share it.
In this video you’ve learned about open data sets, their role in data science, and
where to find them.
We’ve also introduced the Community Data License Agreement, which makes it easier to
share open data.
One important aspect that we didn’t cover in this video is data quality and accuracy,
which might vary greatly depending on who collected and contributed the data set.
While some open data sets might be good enough for personal use, they might not meet enterprise
requirements due to the impact they might have on the business.
In the next module, you will learn about the Data Asset eXchange, a curated open data repository.

Video3.4. Despite the growth of open data sets that are available to the public, it can still

be difficult to discover data sets that are both high quality and have clearly defined
license and usage terms.
To help solve this challenge, IBM created the Data Asset eXchange, or "DAX,”, which
we’ll introduce in this video.
DAX provides a trusted source for finding open data sets that are ready for to use in
enterprise applications.
These data sets and which cover a wide variety of domains, including images, video, text,
and audio.
Because DAX provides a high level of curation for data set quality, as well as licensing
and usage terms, DAX data sets are typically easier to adopt, whether in research or commercial
projects.

Wherever possible, DAX aims to make data sets available under one of the variants of the CDLACommunity Data License Agreement, in order to foster data sharing and collaboration.
DAX also provides a single place to access unique data sets, in particular from IBM Research projects.
To make it easier for developers to get started with using the data sets, DAX also provides tutorials in the form of notebooks that walk through the basics of data cleaning, pre-processing, and exploratory analysis.
For some data sets, there are also notebooks illustrating how to perform more complex analysis, such as creating charts, statistical analysis, time-series analysis, training machine learning models, and integrating deep learning via using the Model Asset eXchange, (a project closely related to DAX and also available on the IBM Developer website).
In this way, DAX helps developers to create end-to-end analytic and machine learning workflows
and to consume open data and models with confidence under clearly defined license terms.
Let's say you've found a data set that might be of interest to you.
On the data set page you can download the compressed data set archive from cloud storage, explore the data set using Jupyter Notebooks, review the data set metadata, such as format, licensing terms and size, and preview some parts of the data set.
Most data sets on DAX are complemented by one or more Jupyter Notebooks that you can use to perform data cleaning, pre-processing, and exploratory analysis.
These notebooks run "as is"as is in Watson Studio, IBM's Data Sciencedata science platform.
Jupyter Notebooks and Watson Studio are covered later during in this course.
In this video, you've learned about IBM's open data repository, the Data Asset eXchange.
In the hands-on lab you'll have a chance to explore the repository.

Video 3.5. In this video, we'll introduce you to machine learning and deep learning models.

Data contains a wealth of information that can be used to solve certain types of problems.
Traditional data analysis approaches, such as a person manually inspecting the data or a specialized computer program that automates the human analysis, quickly reach their limits due to the amount of data to be analyzed or the complexity of the problem.
Machine learning uses algorithms – also known as "models" - to identify patterns in the data.
The process by which the model learns these patterns from data is called "model training."
Once a model is trained, it can then be used to make predictions.
When the model is presented with new data, it tries to make predictions or decisions based on the patterns it has learned from past data.
Machine learning models can be divided into three basic classes: supervised learning, unsupervised learning, and reinforcement learning.
Supervised learning is one of the most commonly used type of machine learning models.
In supervised learning, a human provides input data and the correct outputs.
The model tries to identify relationships and dependencies between the input data and the correct output.
Generally speaking, supervised learning is used to solve regression and classification problems.
Let's look at an example for each problem type:
Regression models are used to predict a numeric, or "real," value.
For example, given information about past home sales, such as geographic location, size, number of bedrooms, and sales price, you can train a model to predict the estimated sales price for other homes with similar characteristics.
Classification models are used to predict whether something belongs to a category, or "class."

For example, given a set of emails along with a designation of whether or not they are considered
spam, an algorithm can be trained to identify unsolicited emails.
In unsupervised learning, the data is not labelled by a human.
The models must analyze the data and try to identify patterns and structure within the
data based only on the characteristics of the data itself.
Clustering and anomaly detection are two examples of this learning style.
Clustering models are used to divide each record of a data set into one of a small number
of similar groups.
An example of a clustering model could be providing purchase recommendations for an
e-commerce store based on past shopping behavior and the content of a shopping basket.
Anomaly detection identifies outliers in a data set, such as fraudulent credit card transactions
or suspicious online log-in attempts.
The third type of learning, reinforcement learning, is loosely based on the way human
beings and other organisms learn.
Think about a mouse in a maze.
If the mouse gets to the end of the maze it gets a piece of cheese.
This is the “reward” for completing a task.
The mouse learns – through trial and error – how to get through the maze to get as
much cheese as it can.
In a similar way, a reinforcement learning model learns the best set of actions to take,
given its current environment, in order to get the most reward over time.
This type of learning has recently been very successful in beating the best human players
in games such as go, chess, and popular strategy video games.
Deep learning is a specialized type of machine learning.
It refers to a general set of models and techniques that tries to loosely emulate the way the
human brain solves a wide range of problems.
It is commonly used to analyze natural language, both spoken and text, as well as images, audio,
and video, to forecast time series data and much more.
Deep learning has had a lot of recent success in these and other areas and is therefore
becoming an increasingly popular and important tool for data science.
Deep learning typically requires very large data sets of labeled data to train a model,
is compute-intensive, and usually requires special purpose hardware to achieve acceptable
training times.
You can build a custom deep learning model from scratch or use pre-trained models from
public model repositories.
Deep learning models are implemented using popular frameworks such as TensorFlow, PyTorch,
and Keras.
Deep learning frameworks typically provide a Python API, and many support other programming
languages, such as C++ and JavaScript.
You can download pre-trained state-of-the-art models from repositories that are commonly
referred to as model "zoos."
Popular model zoos include those provided by TensorFlow, PyTorch, Keras, and ONNX.
Models are also published by academic and commercial research groups.
While it is beyond the scope of this video to explain in detail how you would go about
building a model, let’s briefly outline the high-level tasks using an example.
Assume you want to enable an application to identify objects in images by training a deep
learning model.
First, you collect and prepare data that will be used to train a model.
Data preparation can be a time-consuming and labor-intensive process.

In order to train a model to detect objects in images, you need to label the raw training
data by, for example, drawing bounding boxes around objects and labeling them.
Next, you build a model from scratch or select an existing model that might be well suited
for the task from a public or private resource.
You then train the model on your prepared data.
During training, your model learns from the labeled data how to identify objects that
are depicted in an image.
Once training has commenced, you analyze the training results and repeat the process until
the trained model performance meets your requirements.
When the trained model performs as desired, you deploy it to make it available to your
applications.
In this video, you've learned about machine learning and deep learning, what they are
used for, and where to find open source models.
In the next video, we'll introduce you to the Model Asset eXchange, a curated collection
of ready-to-use and customizable deep learning models.

Video3.6. In this video, we will introduce you to the Model Asset eXchange on IBM Developer, a free

open source resource for deep learning models.
Throughout the video we will refer to the Model Asset eXchange as "MAX."
In the previous video, we briefly outlined the high-level tasks you need to complete
to train a model from scratch.
Due to the amount of data, labor, time, and resources required to complete the tasks,
time to value can be quite long.
To reduce time to value, consider taking advantage of pre-trained models for certain types of
problems.
These pre-trained models can be ready to use right away, or they might take less time to
train.
The Model Asset eXchange is a free open source repository for ready-to-use and customizable
deep learning microservices.
These microservices are configured to use pre-trained or custom-trainable state-of-the-art
deep learning models to solve common business problems.
These models have been reviewed, tested, and can be quickly deployed in local and cloud
environments.
All models in MAX are available under permissive open source licenses, making it easier to
use them for personal and commercial purposes and reducing the risk of legal liabilities.
On MAX, you can find models for a variety of domains, including image, audio, video,
and natural language analysis.
This list includes a small selection.
In the lab for this module, you'll have a chance to explore those models.
Let's take a look at the components of a typical model-serving microservice.
Each microservice includes the following components:
A pre-trained deep learning model.
Code that pre-processes the input before it is analyzed by the model and code that post-processes
the model output.
A standardized public API that makes the services' functionality available to applications.
The MAX model-serving microservices are built and distributed as open-source Docker images.
Docker is a container platform that makes it easy to build applications and to deploy
them in a development, test, or production environment.
The Docker image source is published on GitHub and can be downloaded, customized as needed,

and used in personal or commercial environments.
You can deploy and run these images in a test or production environment using Kubernetes,
an open-source system for automating deployment, scaling, and management of containerized applications
in private, hybrid, or public clouds.
A popular enterprise-grade Kubernetes platform is Red Hat OpenShift, which is available on
IBM Cloud, Google Cloud Platform, Amazon Web Services, and Microsoft Azure.
The model-serving microservices expose a REST API that developers can use to incorporate
deep learning into their applications and services.
Because REST APIs can be consumed using any programming language, you can easily integrate
these services into your existing ecosystem.
The API exposes a prediction endpoint and one or more metadata endpoints.
This example shows the endpoints for the Object Detection microservice.
The /model/predict endpoint takes an image as input and returns as a response a list
of objects that were detected in the image, along with bounding box coordinates that identify
where the detected object is located.
Some prediction endpoints can also accept additional input parameters that impact the
produced results, such as filters.
This microservice exposes two metadata endpoints, /model/labels and /model/metadata.
These endpoints provide information such as the objects that can be detected and the deep
learning model used to derive the answer given the input.
In the lab portion of this module, you will have a chance to explore and test these endpoints
using a web browser.
Each endpoint accepts application-friendly inputs, such as an image in JPG, PNG, or GIF
format, instead of a model-specific data structure.
Each endpoint also generates application-friendly outputs, such as standardized JSON, which
is a lightweight data-interchange format.
Let's take a closer look at what happens when an application invokes the prediction
endpoint.
In this example, a user has selected an image in a web application, the prediction endpoint
is invoked, and the image is uploaded.
The microservice prepares the input image for processing, runs the deep learning model
that identifies objects in the image, generates a response using the prediction results, and
returns the result to the application.
The application renders the results by drawing bounding boxes and labels.
In this video, we've introduced the Model Asset eXchange, a free and open source repository
for microservices that make deep learning functionality available to applications and
services in local and cloud environments.
In the lab, you will have a chance to try a model-serving microservice, explore its
API, and learn more about how you can leverage it from a web application and an Internet
of Things application.

Reading