Libraries for Data Science

Learning Objectives

- Understand various libraries for scientific computing in Python.
- Identify visualization libraries in Python.
- Recognize high-level machine learning and deep learning libraries.
- Explore libraries used in other programming languages.

Overview of Libraries

- **Libraries**: Collections of functions and methods enabling users to perform various actions without writing extensive code.

Key Python Libraries

1. Scientific Computing Libraries

- **Pandas**
  - Offers data structures and tools for effective data cleaning, manipulation, and analysis.
  - Key component: **Data Frame** (a two-dimensional table with rows and columns).
  - Provides easy indexing for data access.
- **NumPy**
  - Focuses on arrays and matrices.
  - Allows application of mathematical functions on arrays.
  - **Pandas** is built on top of **NumPy**.

2. Visualization Libraries

- **Matplotlib**
  - Most well-known library for data visualization.
  - Used for creating customizable graphs and plots.
- **Seaborn**

    - Based on Matplotlib.
    - Generates heat maps, time series, and violin plots.

3. High-Level Machine Learning Libraries

- **Scikit-learn**
    - Contains tools for statistical modeling: regression, classification, clustering, etc.
    - Built on **NumPy**, **SciPy**, and **Matplotlib**.
    - Simple to get started with a focus on model definition and parameter specification.

4. High-Level Deep Learning Libraries

- **Keras**
    - Facilitates building standard deep learning models.
    - High-level interface allows quick and simple model creation.
    - Can utilize GPU but may require a lower-level framework for certain tasks.
- **TensorFlow**
    - A low-level framework for large-scale production of deep learning models.
    - Designed for production and deployment; can be complex for experimentation.
- **PyTorch**
    - Preferred for experimentation.
    - Simplifies the testing of ideas for researchers.

5. General-Purpose Data Processing Framework

- **Apache Spark**
    - A cluster-computing framework for processing data using compute clusters.
    - Enables parallel processing across multiple computers.
    - Comparable functionality to **Pandas**, **NumPy**, and **Scikit-learn**.
    - Supports multiple programming languages: Python, R, Scala, and SQL.
- **Vegas**
    - A Scala library for statistical data visualizations.
    - Works with both data files and Spark Data Frames.

6. Libraries in R

- **ggplot2**
    - Popular library for data visualization in R.
    - Offers built-in functionalities for machine learning and data visualization.
    - Interfaces with Keras and TensorFlow.

Conclusion

- Python libraries are becoming the standard in open-source data science, surpassing R.
- Libraries streamline the process of data analysis and modeling, allowing for efficient communication of results through visualization.

Application Program Interfaces (API)

Learning Objectives

- Define an API.
- List API libraries.
- Define REST API concerning request and response.

Definition of API

- **Application Programming Interface (API)**: A set of rules and protocols that allows different software components to communicate with each other.
    - Enables communication between two software systems using inputs and outputs without needing to understand the backend processes.

Example: Pandas Library

- **Pandas**: A software library where not all components are necessarily written in Python.
    - You can use the **Pandas API** to process data by communicating with other software components.

Backend Flexibility

- The backend of APIs can be written in different programming languages.
  - **TensorFlow**: Backend example written in C++, with APIs available for:
    - Python
    - JavaScript
    - Java
    - Go
  - Other languages with volunteer-developed APIs for TensorFlow include:
    - Julia
    - Matlab
    - R
    - Scala

REST APIs

- **REST**: Stands for **Representational State Transfer**.
  - REST APIs allow communication over the internet, leveraging resources like storage, data, and AI algorithms.

Client-Server Communication

- In REST APIs, your program acts as the **client**, and the API communicates with a **web service**.
  - **Endpoint**: The client finds the service through an endpoint.
  - The client sends **requests** to the resource and receives **responses**.

HTTP Communication

- Data is transmitted using **HTTP methods**.
  - Requests sent to REST APIs typically include a **JSON file** with instructions for the operation to be performed.
  - The web service processes the request and returns a response in the form of a JSON file.

Examples of REST APIs

1. **Watson Text to Speech API**
    a. Converts speech to text.
    b. Example of operation:
        i. The client sends an audio file (POST request).
        ii. The API returns the text transcription of the audio.
        iii. At the backend, the API performs a **GET request** to retrieve the transcription.
2. **Watson Language Translator API**
    a. Translates text (e.g., English to Spanish).
    b. Example of operation:
        i. The client sends the text to be translated.
        ii. The API processes the request and returns the translated text.

Conclusion

- An **API** facilitates communication between software components.
- The **Pandas API** is a user-facing part of the Pandas library.
- **REST APIs** enable internet communication, allowing access to various resources and functionalities.

Overview of IBM Data Asset Exchange (DAX)

- **Purpose**:
    - DAX helps users navigate and explore open data sets, providing a repository of high-quality data suitable for enterprise applications.

Features of DAX

- **Curated Collection**:
    - Contains open data sets from IBM research and trusted third-party sources.
    - Data sets available in various formats including images, video, text, and audio.
- **Data License**:
    - Datasets are available under the **Community Data License Agreement (CDLA)**, ensuring clear usage terms.

- **Access to High-Quality Data**:
    - DAX simplifies the process of finding unique and reliable data sets for developers.

Tutorial Notebooks

- DAX provides tutorial notebooks that cover:
    - **Basic Tasks**: Data cleaning, preprocessing, and exploratory analysis.
    - **Advanced Tasks**:
        - Creating charts.
        - Training machine learning models.
        - Integrating deep learning via the Model Asset Exchange.
        - Running statistical and time-series analysis.

Resources for Developers

- Both the **Data Asset Exchange (DAX)** and **Model Asset Exchange (MAX)** are accessible through the IBM Developer Website, enabling developers to:
    - Create end-to-end analytic and machine learning workflows.
    - Consume open data and models confidently under defined license terms.

Example Data Set

- **NOAA Weather Data**:
    - Example of a data set available on DAX.
    - Contains data from a weather station at JFK Airport in New York.

Interacting with Data Sets on DAX

- **Data Set Page**:
    - Users can click to download datasets from cloud storage.
    - Run associated notebooks in Watson Studio for data exploration and analysis.
    - Preview metadata and glossary related to the dataset.
- **Assets Tab**:
    - View all Jupyter notebooks and data related to the chosen dataset.
    - Execute notebooks in Watson Studio for various analyses.
- **Data Files**:

    - Users can view available data files within their projects by clicking the data option.

Key Takeaways

- DAX offers high-quality open data sets and tutorial notebooks for developers.
- Provides a centralized platform for accessing and utilizing open data for enterprise applications.
- Facilitates the process of data cleaning, analysis, and machine learning model development.

Notes on Machine Learning Models

Introduction

Welcome to “Machine Learning Models – Learning from models to make predictions.” By the end of this video, you’ll be equipped to define what a machine learning model is, describe the different types of learning models, and understand how to use these models to tackle problems.

The Importance of Data

Data is rich with information that can help us solve various problems. Traditionally, data analysis involved either a person manually inspecting data or using specialized computer programs for automation. However, these methods can struggle with the vast amount of data and the complexity of certain problems.

What is Machine Learning?

Machine learning (ML) comes into play here. It uses algorithms—often referred to as “models”—to identify patterns in data. The process through which a model learns these patterns is known as “model training.”

Making Predictions with Trained Models

Once a model is trained, it can be used to make predictions. When new data is fed into the model, it utilizes the patterns it learned from past data to make informed predictions or decisions.

Types of Machine Learning Models

Machine learning models can be categorized into three main classes:

1. **Supervised Learning**
    a. In this approach, humans provide both input data and the corresponding correct outputs.
    b. The model seeks to understand the relationships and dependencies between the input data and the correct output.
    c. **Two main types of supervised models**:
        i. **Regression Models**: Used for predicting numeric values. For instance, given data on past home sales—like location, size, and number of bedrooms—you can train a model to estimate the sales price of similar homes.
        ii. **Classification Models**: These models predict whether certain data belongs to a particular category. A common example is classifying emails as spam or not.
2. **Unsupervised Learning**
    a. Here, the data is unlabeled. The models analyze the data to find patterns and structures based solely on its characteristics.
    b. **Example**: Clustering models group similar records together. For instance, an e-commerce platform might use clustering to recommend products based on past shopping behaviors.
3. **Reinforcement Learning**
    a. This type of learning is inspired by how humans and animals learn through trial and error. Imagine a mouse navigating a maze: when it reaches the end, it gets a reward (cheese).
    b. The model learns the best actions to take in a given environment to maximize rewards over time. This method has led to breakthroughs in games like Go and chess.

Deep Learning

Deep learning is a specialized subset of machine learning that mimics how the human brain operates to solve a wide variety of problems. It's particularly effective in analyzing natural language, images, audio, and video.

- **Key Points**:

- Requires large datasets of labeled data for training.
- Is compute-intensive and typically needs specialized hardware to train models efficiently.
- Popular frameworks for deep learning include TensorFlow, PyTorch, and Keras.

Building a Model: A Quick Overview

Let’s say you want to develop an application to identify objects in images. Here’s a high-level view of the process:

1. **Data Collection and Preparation**: You’ll need to gather raw data and label it appropriately (e.g., drawing bounding boxes around objects).
2. **Model Selection**: Decide whether to build a model from scratch or select a suitable existing one from a public resource.
3. **Training the Model**: Train your model using the prepared data. It will learn to identify objects based on the labels you provided.
4. **Analyzing Results**: Monitor the training process and refine it until the model meets your performance requirements.
5. **Deployment**: Once satisfied with the model's performance, deploy it for use in applications.

Conclusion

In summary, machine learning leverages algorithms to identify data patterns. The primary types include supervised, unsupervised, and reinforcement learning. Supervised learning further divides into regression and classification, while deep learning represents a complex yet powerful approach.

Here are the notes in a more human-like style based on the transcript about the Model Asset eXchange (MAX):

Notes on the Model Asset eXchange (MAX)

Introduction

Welcome to the Model Asset eXchange (MAX)! After watching this video, you'll be able to navigate the MAX from IBM and understand how deep learning model-serving detects images.

What is the Model Asset eXchange?

- **Overview**: MAX is a free, open-source resource for deep learning models available on the IBM Developer platform.
- **Challenges of Training Models**: Training a model from scratch can be resource-intensive, requiring large datasets, significant time, and labor. This often leads to a longer time to value.
- **Solution**: To expedite this process, consider using pre-trained models that can be immediately utilized or require less training time.

Model Creation Process

- Models are developed by running data through them using computational resources and domain expertise.
- The typical steps involved are:
    - **Research**
    - **Evaluation**
    - **Testing**
    - **Training**
    - **Validation**
- Following these steps results in a validated model ready for deployment.

Features of the Model Asset eXchange

- MAX serves as a repository for ready-to-use and customizable deep learning microservices.
- These microservices are designed to utilize either pre-trained models or custom trainable models to address common business challenges.

- All models available on MAX are fully tested, ensuring they can be quickly deployed in local or cloud environments.
- The models are distributed under permissive open-source licenses, minimizing legal risks for personal or commercial use.

Types of Models Available

You can find various models in different domains on MAX, including:

- Object detection
- Image, audio, and video classification
- Text classification
- Named entity recognition
- Image-to-text translation
- Human pose detection

Components of a Model-Serving Microservice

A typical model-serving microservice includes:

- **Pre-trained Deep Learning Model**: The core model used for analysis.
- **Pre-processing Code**: Prepares input data for analysis by the model.
- **Post-processing Code**: Manages the model output after analysis.
- **Standardized Public API**: Allows applications to access the microservice's functionality.

Building and Deploying Microservices

- Model-serving microservices operate by running inputs through a validated model, then applying the output via a REST API.
- Steps for development include implementing, packaging, documenting, and testing the microservice.
- Once complete, the microservice can be deployed to local machines or private/hybrid/public clouds.

Docker and Kubernetes Integration

- MAX microservices are distributed as open-source Docker images, facilitating easy building and deployment of applications.
- The source code for these Docker images is published on GitHub and can be customized for various uses.
- To automate deployment, scaling, and management of these Docker images, you can use Kubernetes.
- **Red Hat OpenShift**: A popular enterprise-grade Kubernetes platform available on various cloud services, including IBM Cloud, Google Cloud Platform, Amazon Web Services, and Microsoft Azure.

Conclusion

In summary, the Model Asset eXchange is a valuable repository for customizable deep learning microservices that can significantly reduce the time to value by leveraging pre-trained models. MAX microservices are built as open-source Docker images and can be efficiently managed using Kubernetes, specifically Red Hat OpenShift for enterprise applications.